

Advanced Software Testing Volume 2

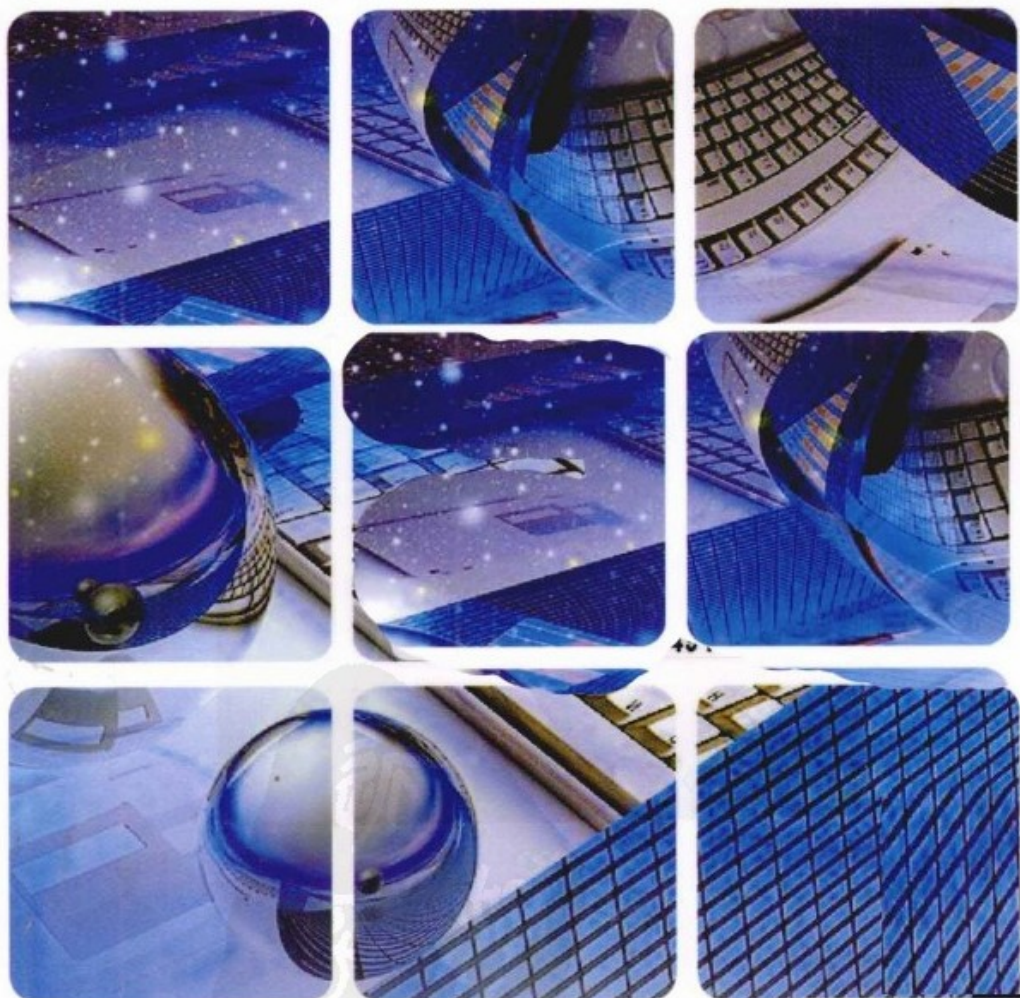
# 高级软件测试

## 高级软件测试经理

### 卷2

Rex Black 著

刘 琴 周震漪  
郑文强 马均飞 熊晓虹 译



和  
PDF

清华大学出版社



# Advanced Software Testing Volume 2

## 内容简介

本书是目前市面上唯一完整的完全按照国际软件测试认证委员会 (ISTQB) 的测试人员认证高级大纲撰写的专业书籍系列之一。本书章节与高级大纲中高级软件测试经理模块的标题、顺序和知识点相匹配, 适宜ISTQB-ATM(高级软件测试经理) 的认证考前学习, 也可以作为软件测试经理的常用指导手册。

本书主要介绍在软件测试估算、策划、监视和控制中, 一个软件测试经理熟练运用高级技能所必须掌握的知识, 包括制定软件测试系统的总体测试目标和测试策略; 计划测试任务和进度, 组织测试活动, 采用各种度量对测试和风险进行评估、报告、跟踪和控制。本书通过大量案例介绍了如何挑选、获取、分配测试任务所需要的充分的资源; 如何组建、管理和领导测试团队, 负责协调测试团队各成员之间以及测试团队和各利益相关者之间的沟通。

本书每一章都提供与ISTQB高级大纲一致的, 充足的认证考试模拟题, 为准备参加ISTQB-ATM资质认证考试的人员提供相应的基础性知识; 每一章节的课堂练习都结合实际案例, 并提供答案, 具有较高的参考价值。

ISBN 978-7-302-26163-6



9 787302 261636 >

定价: 43.00元



Advanced Software Testing Volume 2

# 高级软件测试

## 高级软件测试经理

### 卷2

Rex Black 著

刘 琴 周震漪  
郑文强 马均飞 熊晓虹 译

清华大学出版社  
北京



Simplified Chinese edition copyright © 2010 by Rock Nook Inc.

Original English language title from Proprietor's edition of the Work.

Original English language title: Advanced Software Testing, Volume 2 by Rex Black © 2010

EISBN: 978-1-933952-36-9

All Rights Reserved.

Published by arrangement with the original publisher, Rock Nook Inc.

This edition is authorized for sale only in the People's Republic of China (excluding the Special Administrative Region of Hong Kong and Macao).

本书中文简体翻译版由 Rock Nook Inc. 授权给清华大学出版社在中国境内(不包括中国香港、澳门特别行政区)出版发行。

北京市版权局著作权合同登记号 图字: 01-2009-7736 号

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话: 010-62782989 13701121933

## 图书在版编目(CIP)数据

高级软件测试·卷2:高级软件测试经理/(美)布莱克(Black, R.)著;刘琴等译. —北京:清华大学出版社, 2012.1

书名原文: Advanced Software Testing, Volume 2,  
ISBN 978-7-302-26163-6

I. ①高… II. ①布… ②刘… III. ①软件—测试 IV. ①TP311.5

中国版本图书馆CIP数据核字(2011)第136019号

责任编辑:龙啟铭

责任校对:李建庄

责任印制:李红英

出版发行:清华大学出版社

地址:北京清华大学学研大厦A座

<http://www.tup.com.cn>

邮编:100084

社总机:010-62770175

邮购:010-62786544

投稿与读者服务:010-62795954, [jsjic@tup.tsinghua.edu.cn](mailto:jsjic@tup.tsinghua.edu.cn)

质量反馈:010-62772015, [zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

印装者:北京鑫海金澳胶印有限公司

经销:全国新华书店

开本:185×260 印张:22

字数:543千字

版次:2012年1月第1版

印次:2012年1月第1次印刷

印数:1~3000

定价:43.00元

---

产品编号:035689-01



# 译者序

作为国际软件测试认证委员会 (ISTQB) 大中华首席代表, 我目睹了祖国软件产业日新月异的发展, 以及从跟随到逐步开始影响国际软件测试技术潮流的趋势。在软件测试行业, ISTQB 在全世界的影响力日益壮大。截至 2011 年 5 月 ISTQB 成员国已经达到 48 个, 全球证书持有者总数已经超过 16 万人。在测试市场逐步向亚太地区扩张的形势下, 如何迅速提高英语为非母语的中国测试从业人员在测试领域的竞争力? 最直接有效的途径是掌握国际通用的测试行业知识。ISTQB 为业界唯一公认的测试知识以及认证体系, 其非营利及开放性不但吸纳融合了国际测试领域知识和经验的精华, 并且确保了该体系的与时俱进。目前, 译者正在参与的撰写中的国际软件与系统测试标准 ISO 29119 也是与 ISTQB 知识体系完全兼容的。因此我们审慎地选择了向知名出版社推荐并引进基于 ISTQB 大纲撰写的基础级和高级系列教材。

2006 年, 在上海静安寺附近的一家小咖啡厅, 本书的四位译者初次汇聚在一起, 从翻译最基本的统一词汇开始, 到目前大中华区分会 (CSTQB) 拥有 30 余位国内专家, 目前已经完全引进 ISTQB 最新知识体系, 中国证书持有者 3000 余人, 有 2 位专家直接参与新版本 ISTQB 高级和专家级编写, 并看到相关翻译教材的出版。回想这一切, 我非常感慨, 激励我们翻译本书的初衷就是为推动国内的软件测试行业的国际化、标准化尽一份微薄之力。虽然这本书是基于 ISTQB 高级大纲第二个模块编写的, 但许多案例还是从美国企业抽取的, 与我国的国情和文化并不十分贴合。我也希望能够协助国内的业界精英, 开发出基于本土案例的教材或者教学课件, 更有效地推动, 甚至参与 ISTQB 及相关知识体系的发展和创新。

本书的作者 Rex Black 先生是 ISTQB 的第三届主席, 也是美国测试委员会的发起人之一, 在测试界深具影响力和号召力。我在翻译本书期间一直和 Black 先生保持沟通, 感谢他不厌其烦地释疑解难, 并坦诚地接受个别译者对原文的疑问和建议。本书紧扣 ISTQB 高级认证大纲中高级软件测试管理者部分, 主要介绍在软件测试估算、策划、监视和控制中一个软件测试经理熟练运用高级技能所必须掌握的知识, 包括制定软件测试系统的总体测试目标和测试策略; 计划测试任务和进度, 组织测试活动, 采用各种度量对测试和风险进行评估、报告、跟踪和控制。本书通过大量案例介绍了如何挑选、



获取、分配测试任务所需要的充分的资源；如何组建、管理和领导测试团队，负责协调测试团队各成员之间以及测试团队和各利益相关者之间的沟通。

本书适用于掌握测试工程、测试设计、测试工具、软件开发生命周期和测试管理的基本概念并希望获得 ISTQB 高级资质认证证书的人群。本书包含了选择管理软件测试作为职业的软件从业者应当掌握的相应内容，着重讲述了和测试分析、测试设计、测试执行和测试结果评估相关的技能和技术。

本书已被 CSTQB (Chinese Software Qualifications Testing Board 国际软件测试认证委员会大中华分会) 指定为官方培训教材。虽然工程师的素养有赖于知识教育、实践能力和企业经验的融合，但通过对本书的学习，能够为高校及企业培养具有国际竞争力的软件测试工程师提供一个可参考的，与 ISTQB 对接的渠道。

在此我谨以 ISTQB 大中华首席代表和 CSTQB 理事长代表的名义，向共同担任翻译工作的周震漪、郑文强、马均飞和熊晓虹表示最诚挚的谢意，同时也要感谢同济大学软件学院的师生花费宝贵时间，与我反复研讨书中的一些经典案例的译文。最后我心怀歉疚地感谢一直支持我的先生徐添和我三岁的女儿徐贝宁，没有你们的理解和支持，我难以想象在我本已繁忙的工作之余还能完成这本书的翻译。

由于水平和时间的限制，书中的翻译难免会出现错误，欢迎读者以及各界同仁不吝指正。

刘琴 教授 博士  
同济大学软件学院常务副院长  
ISTQB 中国首席代表  
CSTQB 理事长代表  
CCF 软件工程专业委员会委员  
2011 年 11 月



# 前言

作为软件工程师，我从事软件测试行业已经超过 25 年。当我还是个程序员的时候，我就学习如何对自己开发的代码进行组件测试。20 年前当我成为一名专业的测试人员之后，我不仅测试自己的代码，也要测试其他人的。

我在 20 世纪 80 年代获得了加州大学洛杉矶分校的软件工程学位，这是一所享誉全球的一流大学。但是，我在那里学到的测试知识非常有限。所以当 I 成为一名专业测试人员的时候，我只好通过阅读书籍和积累项目经验来自学并掌握必需的技能。

在 20 世纪 80 年代，软件测试只是软件工程领域刚刚兴起的一个独立分支。因此在高等学府甚至一流大学都没有提供软件测试相关的教育课程，那也就不足为奇了。

不幸的是，在这此后的 20 年，大学和学院中关于软件测试的教育都没有获得很大的改善。一些私营公司提供的软件测试培训，包括我自己的公司 RBCS，帮助填补了市场空白。但是，测试人员获得的培训通常都是断断续续的，没有对特定方面进行连续的培训。因此，许多软件测试的从业者还是通过自学或者非系统学习方法来帮助自己理解整个测试领域。这样造成了很多的测试从业者甚至连软件测试的基础和高级的概念都搞不清楚。

国际软件测试认证委员会 (ISTQB) 通过定义一组大纲 (或者知识体系，如果您喜欢这么理解的话) 来指导从业者在其通往软件测试专家的职业发展道路上应当掌握哪些相应的内容。我曾经担任国际软件测试认证委员会和美国软件测试认证委员会，我也曾就职于 ISTQB 基础级工作小组和高级工作小组制定这些大纲。

之前我曾经和 Isabel Evans、Dorothy Graham、Erik van Veenendaal，合著过一本书，叫做 “Foundations of Software Testing”，书中介绍了软件测试基础级大纲。

本书中讨论的软件测试高级内容是针对测试经理的。这里，我会讲述测试从业者在职业生涯中达到软件测试高级水平所要掌握并应用的一些概念。我希望通过阅读本书，你们会觉得在理解和应用这些概念的时候更加得心应手。在未来的 20 年里，在我从这个领域退休之前，我希望提高人们对软件测试零散、片面的初步理解，而使其成为一门真正受人尊敬的专业性行业。我希望这本书可以帮助你们成为这些受人尊敬的测试从业者中的一员。

# 目 录

第 1 章 软件测试基础	1
1.1 概述	1
1.2 软件生命周期中的测试	2
1.3 顺序生命周期模型	3
1.4 迭代或增量生命周期模型	4
1.5 螺旋生命周期模型	5
1.6 测试级别	6
1.6.1 两个将测试集成到生命周期的案例研究	8
1.6.2 软件生命周期中的测试练习	9
1.6.3 软件生命周期中的测试练习参考答案	9
1.7 特定系统	12
1.7.1 综合系统项目案例研究	14
1.7.2 安全关键系统	15
1.8 度量元和度量	16
1.8.1 度量元和度量的练习	18
1.8.2 度量元和度量的练习参考答案	18
1.9 职业道德	19
1.10 认证考试模拟题	20
第 2 章 测试过程	22
2.1 概述	22
2.2 测试过程模型	23
2.3 测试计划和控制	23
2.3.1 测试计划 and 控制的案例研究	25
2.4 测试分析和设计	26
2.5 测试实施和执行	27
2.5.1 测试执行	28
2.5.2 测试执行前置条件的案例学习	29
2.5.3 测试标准 BS 7925/2	31



2.6	评估出口准则和报告 .....	32
2.7	测试结束活动 .....	35
2.7.1	测试结束的两个案例 .....	36
2.7.2	测试结束活动练习 .....	38
2.7.3	测试结束活动练习参考答案 .....	38
2.8	认证考试模拟题 .....	38
<b>第3章</b>	<b>测试管理 .....</b>	<b>40</b>
3.1	概述 .....	41
3.2	基于风险的测试与失效模式和影响分析 .....	41
3.2.1	基于风险测试的特性和好处 .....	43
3.2.2	基于风险测试的历史 .....	45
3.2.3	如何进行基于风险的测试 .....	46
3.2.4	风险级别 .....	48
3.2.5	控制风险 .....	49
3.2.6	项目风险 .....	50
3.2.7	两种工业标准以及它们与风险的关系 .....	52
3.2.8	基于风险的测试与失效模式和影响分析练习 1 .....	53
3.2.9	基于风险的测试与失效模式和影响分析练习 1 参考答案 .....	54
3.2.10	风险识别和评估技术 .....	56
3.2.11	质量风险分类 .....	57
3.2.12	记录质量风险 .....	59
3.2.13	用 ISO 9126 进行质量风险分析 .....	61
3.2.14	用风险发生成本进行质量风险分析 .....	62
3.2.15	用危害分析进行质量风险分析 .....	63
3.2.16	判定所有风险的优先级 .....	63
3.2.17	利益相关者参与 .....	65
3.2.18	基于风险的测试与失效模式和影响分析练习 2 .....	65
3.2.19	基于风险的测试与失效模式和影响分析练习 2 参考答案 .....	66
3.2.20	失效模式和影响分析 (FMEA) .....	70
3.2.21	用失效模式和影响分析进行质量风险分析 .....	71
3.2.22	决定风险优先级数字 .....	73
3.2.23	FMEA 的收益、成本和挑战 .....	73
3.2.24	FMEA 案例学习 .....	74
3.2.25	基于风险的测试与失效模式和影响分析练习 3 .....	74
3.2.26	基于风险的测试与失效模式和影响分析练习 3 参考答案 .....	76
3.2.27	基于风险的测试和测试过程 .....	81
3.2.28	整个生命周期中的基于风险的测试 .....	82

3.2.29	基本测试过程中的基于风险的测试	84
3.2.30	基于风险测试的挑战	86
3.2.31	基于风险的测试与失效模式和影响分析练习 4	88
3.2.32	基于风险的测试与失效模式和影响分析练习 4 参考答案	89
3.2.33	FMEA 案例学习二	90
3.2.34	基于风险的测试和测试控制	92
3.2.35	基于风险的测试结果评估和报告	93
3.2.36	基于风险的测试与失效模式和影响分析练习 5	95
3.2.37	基于风险的测试与失效模式和影响分析练习 5 参考答案	95
3.3	测试管理文档和测试计划文档模板	97
3.3.1	测试方针文档	98
3.3.2	测试策略文档	99
3.3.3	测试策略类型	101
3.3.4	测试计划模板	103
3.3.5	IEEE 829 软件测试文档标准以及它们如何与测试计划文档相关联	104
3.3.6	主测试计划文档	108
3.3.7	级别测试计划文档	109
3.3.8	测试计划和测试计划偏离案例学习	109
3.3.9	主测试计划案例学习	111
3.3.10	一个简单 PC 应用程序的测试计划案例学习	111
3.3.11	测试管理文档和测试计划文档模板练习	114
3.3.12	测试管理文档和测试计划文档模板练习参考答案	114
3.4	测试估算	117
3.4.1	影响估算的因素	118
3.4.2	估算技术	120
3.4.3	使用工业平均值	122
3.4.4	测试点分析	123
3.4.5	协商和减少测试范围	126
3.4.6	测试估算练习	126
3.4.7	测试估算练习参考答案	127
3.5	测试计划安排	132
3.5.1	尽早进行测试计划的案例学习	134
3.6	测试进程监控	136
3.6.1	产品风险度量	137
3.6.2	缺陷度量元	137
3.6.3	测试用例（或规程）度量	140
3.6.4	测试进度监控练习 1	146
3.6.5	测试进度监控练习 1 参考答案	147

3.6.6	测试覆盖率度量	148
3.6.7	风险覆盖率	149
3.6.8	度量的使用	150
3.6.9	两个测试报告的案例学习	153
3.6.10	测试进度监控练习 2	155
3.6.11	测试进度监控练习 2 参考答案	156
3.7	测试的商业价值	158
3.7.1	质量成本	159
3.7.2	测试的其他价值	161
3.7.3	测试商业价值的练习	162
3.7.4	测试商业价值练习答案	162
3.8	分布式、外包、内包测试	163
3.8.1	特殊的分布式、外包和内包测试问题	165
3.8.2	能力成熟度模型集成 (CMM) 和测试	167
3.8.3	分布式测试的案例研究	168
3.9	测试管理问题	169
3.9.1	管理探索性测试的案例研究	173
3.9.2	综合系统问题	174
3.9.3	安全关键系统问题	175
3.10	非功能性测试问题	176
3.10.1	工具和硬件需求	178
3.10.2	公司和安全性考虑	180
3.11	认证考试模拟题	181
第 4 章	测试技术	188
第 5 章	软件特性测试	191
第 6 章	评审	193
6.1	概述	193
6.2	评审的原则	194
6.2.1	正式和非正式评审	195
6.2.2	非正式评审的实例研究	196
6.3	评审的类型	197
6.4	引入评审	199
6.4.1	评审中缺陷消除的有效性	200
6.4.2	两个评审实例研究	201
6.4.3	引入评审练习 1	203
6.4.4	引入评审练习 1 的参考答案	204



6.4.5 引入评审练习 2 .....	204
6.4.6 引入评审练习 2 的参考答案 .....	204
6.5 评审的成功因素 .....	207
6.5.1 回顾一个早期用例研究 .....	209
6.5.2 评审的 IEEE 1028 标准 .....	210
6.6 认证考试模拟题 .....	211
<b>第 7 章 事件管理 .....</b>	<b>212</b>
7.1 概述 .....	212
7.2 何时可以发现一个缺陷 .....	213
7.3 缺陷生命周期 .....	213
7.3.1 缺陷生命周期练习 .....	215
7.3.2 缺陷生命周期练习参考答案 .....	215
7.4 缺陷域 .....	216
7.4.1 缺陷域练习 .....	220
7.4.2 缺陷域联系报告 .....	221
7.5 度量元和事件管理 .....	222
7.5.1 度量元和事件管理练习 .....	224
7.5.2 度量元和事件管理练习参考答案 .....	224
7.6 沟通事件 .....	225
7.7 认证考试模拟题 .....	225
<b>第 8 章 标准以及测试过程改进 .....</b>	<b>227</b>
8.1 概述 .....	227
8.2 需要考虑的标准 .....	228
8.3 测试改进过程 .....	232
8.4 改进测试过程 .....	234
8.4.1 一个通用的过程改进框架 .....	235
8.4.2 案例研究: 测试评估的结果 .....	236
8.5 用 TMM 改进测试过程 .....	237
8.6 用 TPI 改进测试过程 .....	240
8.7 用 CTP 改进测试过程 .....	244
8.8 用 STEP 改进测试过程 .....	248
8.9 能力成熟度模型集成 (CMMI) .....	250
8.10 测试改进过程练习 .....	252
8.11 测试改进过程练习参考答案 .....	252
8.12 认证考试模拟题 .....	254

第9章 测试工具和自动化 .....	256
9.1 概述 .....	256
9.2 测试工具概念 .....	256
9.2.1 测试自动化的成本 .....	258
9.2.2 测试自动化的风险 .....	259
9.2.3 测试自动化的益处 .....	260
9.2.4 测试自动化的策略 .....	261
9.2.5 测试自动化策略的案例分析 .....	262
9.2.6 测试工具集成和脚本 .....	263
9.2.7 集成化测试工具的案例分析 .....	264
9.2.8 测试工具分类 .....	266
9.3 测试工具种类 .....	267
9.3.1 测试管理工具 .....	268
9.3.2 测试执行工具 .....	268
9.3.3 关键字驱动的自动化测试执行 .....	269
9.3.4 测试执行目标的案例分析 .....	270
9.3.5 调试和排错工具 .....	271
9.3.6 故障散播和故障注入 .....	271
9.3.7 静态分析工具 .....	272
9.3.8 动态分析工具 .....	272
9.3.9 性能测试工具 .....	272
9.3.10 网站工具 .....	273
9.3.11 模拟器和仿真器 .....	274
9.3.12 自定义工具开发的案例分析 .....	274
9.4 认证考试模拟题 .....	275
第10章 个人技能和团队构成 .....	277
10.1 概述 .....	277
10.2 个人技能 .....	278
10.2.1 测试技能 .....	279
10.2.2 技术和软件技能 .....	280
10.2.3 用户、业务以及领域技能 .....	280
10.2.4 技能清单和管理 .....	281
10.2.5 个人技能练习 .....	284
10.2.6 个人技能练习参考答案 .....	285
10.3 动态测试团队 .....	288
10.3.1 动态测试团队练习 .....	294

10.3.2	动态测试团队练习参考答案	294
10.4	组织的测试选择	295
10.4.1	混合使用不同独立性的方法	298
10.4.2	外包独立测试	299
10.4.3	混合的质量保证方法的案例分析	299
10.5	激励	301
10.5.1	度量元和激励	302
10.5.2	激励和负面激励评论的案例分析	302
10.6	沟通	303
10.7	认证考试模拟题	304
第 11 章	认证考试准备	307
11.1	学习目标	307
11.1.1	级别 1: 牢记 (K1)	307
11.1.2	级别 2: 理解 (K2)	308
11.1.3	级别 3: 应用 (K3)	308
11.1.4	级别 4: 分析 (K4)	308
11.1.5	学习目标级别的由来	309
11.2	ISTQB 高级认证考试	309
11.2.1	基于场景分析的考题	310
11.2.2	考题的演变	311
附录 A	参考书目	313
附录 B	HELLOCARMS 下一代房屋净值借贷系统	316
附录 C	模拟题答案	326



# 软件测试基础

“噢，我们不会在测试的时候喝酒。”

“对了，Larry，我们喝酒是因为我们在做测试！”

——这是两个大型测试团队总监对我讲的一个互联网系统  
启动测试团队在测试期间喝啤酒的故事的反应。

高级大纲第1章主要介绍和后续章节相关的上下文和背景资料，共有5个部分：

1. 概述。
2. 软件生命周期中的测试。
3. 特定系统。
4. 度量元和度量。
5. 职业道德。

下面依次介绍以上各部分，了解它们和测试管理的关系。

## 1.1 概 述

### 学习目标：

能回想起该部分内容即可。

顾名思义，本章会介绍软件测试的一些基本内容。对软件测试专家而言，掌握这些测试的基本内容和中心思想是非常有必要的。

### ISTQB 术语

**软件生命周期：**软件产品从构思开始到其不再可用的这段时间。软件生命周期通常包括概念阶段、需求阶段、设计阶段、实现阶段、测试阶段、安装和检验阶段、运行和维护阶段，有时候还包括退役阶段。需要注意的是，以上这些阶段可能会重叠或者不断迭代。

本章主要包括以下4个方面：

- 软件生命周期和它们对测试的影响。

- 特定类型系统和它们对测试的影响。
- 测试和质量的度量元和度量。
- 职业道德。

上述许多概念会在后续章节中进行详细说明。您可能认为这些内容是多余的，因为基础级大纲中对以上内容也进行了讲解。实际上这并不多余，而是对基础级大纲内容的合理扩展。

## 1.2 软件生命周期中的测试

### 学习目标:

(K2) 介绍测试如何成为软件开发及维护活动的一部分。

(K4) 分析软件生命周期模型，列出最适合的需要执行的任务和测试活动，区分测试和开发活动。

ISTQB 基础级大纲的第 2 章介绍了如何将测试集成到软件生命周期中。和基础级大纲一样，在高级大纲中我们应该明确的是将测试集成到软件生命周期以确保项目成功。不管选择哪一种生命周期模型，顺序模型、增量模型、迭代模型，还是螺旋模型，这个论断都是正确的。

在软件生命周期中，测试过程和其他过程恰当地融合是成功的关键。这种融合在测试与其他过程活动的关键接口及相互切换中显得尤为必要，例如以下这些过程活动：

- 需求工程和管理。
- 项目管理。
- 配置管理和变更管理。
- 软件开发和维护。
- 技术支持。
- 技术文档。

让我们来看两个关于“联盟 (Alignment)”的例子。

在顺序生命周期模型中，一个重要的假定是项目团队在项目早期定义了软件需求，然后在后续项目过程中管理需求变更（一般是有限的变更）。在这种情况下，如果项目团队遵循规范的需求过程，那么负责系统测试级别的独立测试团队就可以遵循基于需求的分析式测试策略。

如果在顺序模型中使用这样的分析式测试策略，那么测试团队会在项目早期进行测试规划和设计，并通过分析需求的规格说明来确定测试条件。策划、分析和设计工作可以识别需求中的缺陷，使测试成为预防性活动。而失效的检测会在软件生命周期的后期，即系统测试执行以后才开始。

然而，如果项目遵循增量生命周期模型，比如基于敏捷方法之一的 Scrum 模型。这时候测试团队在项目的早期便不会获得完整的需求。而是以 30 天为一个周期，即所谓每个“Sprint”周期开始时获得项目的部分需求。

在这种情况下，测试团队采用的最好策略是识别主要的质量风险域并进行优先级排序，而不是在项目开始的时候才进行需求分析，即测试团队可以采用基于风险分析的测试策略。基于此测试策略，仅在测试执行前才进行针对性的测试设计和开发，速度虽快却潜在地降低了测试的缺陷预防效应。对缺陷的检测则在项目早期，在第一个 Sprint 末期就开始了，并持续贯穿在其后的每一个短的子周期末期。在这种情况下，主测试过程的测试活动会彼此重叠进行，并与软件生命周期中的其他主要活动同时发生。

对测试来讲，无论采用何种软件生命周期模型，尤其是节奏较快的敏捷型软件生命周期模型，有效的需求和配置管理都是至关重要的，缺乏恰当的需求管理将导致测试团队无法了解系统是什么，系统能做什么；而缺乏好的配置管理，正如在基础级大纲中论述过的，将会导致一些需求变更的缺失，以致相关人员将无法及时了解什么时候测试过什么，也就很难透彻地理解测试的结果。

在这里，让我们快速地浏览一下大纲中提到的几种生命周期模型：顺序模型、增量模型、迭代模型以及螺旋模型。

### 1.3 顺序生命周期模型

图 1-1 是顺序模型中的 V 模型，它被称为顺序模型是因为整个系统是一次性构筑而成的，所有的活动都以一定的顺序进行排列定义、实现和测试。最常见的两个例子是瀑布模型和 V 模型。

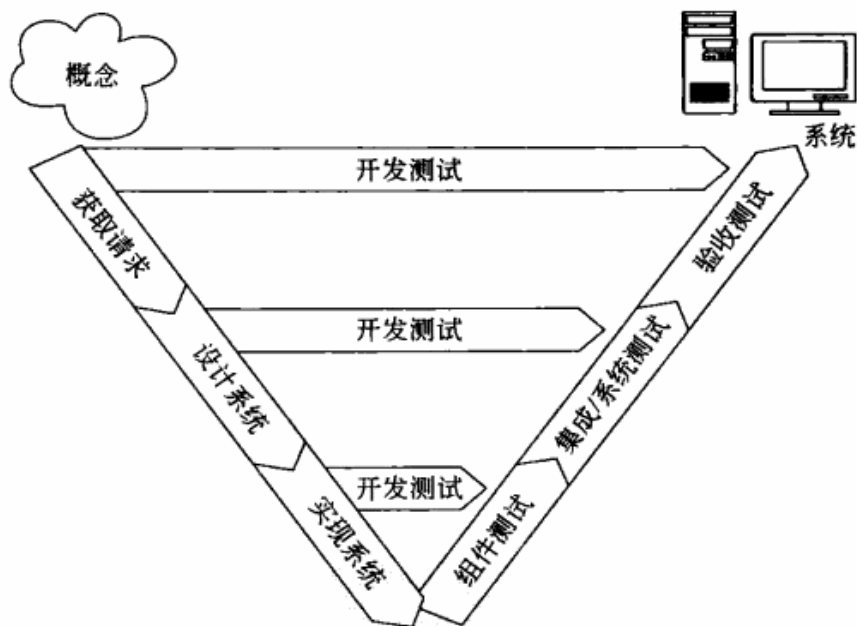


图 1-1 V 模型

您可能也听说过 W 模型，W 模型描绘了各测试级别的测试开发活动，这些行为在 V 模型中排列成 V 型，而在 W 模型中，和 V 模型平行的节点上排列的需求、设计和实现活动等节点和 V 模型原有节点排列成 W 型。W 模型中紧邻测试活动的平行框清楚描述了该测试级别的缺陷修正活动。有人认为 W 模型比 V 模型表达得更加清晰，但是我个人觉得 W 模型的图标让人困惑，因为它比 V 模型看起来更加复杂。



顺序生命周期模型的使用给测试带来了一些需要测试经理管理和解决的问题：第一个也是最让人头疼的问题，即在项目末期的测试过程中如何压缩测试进度。这样的“紧缩”，往往意味着在永远繁忙的项目末期还要在保证软件按期交付之间做无奈地折中。当测试经理被放在质量警察这样的位置，即负责批准软件发布，当某版本发布后，被检测出遍布缺陷，由于测试经理主观承受着各方的巨大压力还是被迫批准其通过，这种做法最终对各利益方都没有好处。

第二个问题是所有开发团队，即交付时间不及时、交付对象不稳定以及交付给测试团队不可测试的系统。这些问题会导致测试进度的重要部分被重复的单元测试所占据。如果没有充足的配置管理，由于上述交付对象不能进行工作，甚至其结构都没有得到很好的理解，高级别的测试变得毫无意义。

第三个问题是没有加入如图 1-1 所示 V 模型图横杠中的活动。由于其他项目的原因或者缺乏完备的管理意识，测试团队在项目后期才加入生命周期，直接导致准备时间不足。这种情况下，测试只是典型地移交一些随机或者最佳反应策略，没有缺陷预防，没有清楚的覆盖率，而且测试价值非常有限。

上述那些顺序生命周期的测试相关问题不是不可避免的，但是需要对它们进行细致地管理。您将会在本书中了解更多如何管理和应对这些问题的方法。

## 1.4 迭代或增量生命周期模型

迭代或增量生命周期模型如图 1-2 所示，该模型分块、迭代地进行系统构建和测试。可以通过基于风险的策略对每一次迭代的功能和性能进行适当分组并构建。这种情况下，最有可能失败的功能和性能会在第一个迭代中构建完成，失败可能性次之的功能和性能会在第二个迭代中完成，以此类推。这样的分组和构建也可以基于客户的优先级，即顾客要求最迫切的功能和性能可以在第一个迭代中构建，要求最弱的可以放在最后的迭代中构建，其余的放在项目中期的迭代中构建。迭代的分组过程也可能会受法规需求、设计需求以及其他条件的约束。

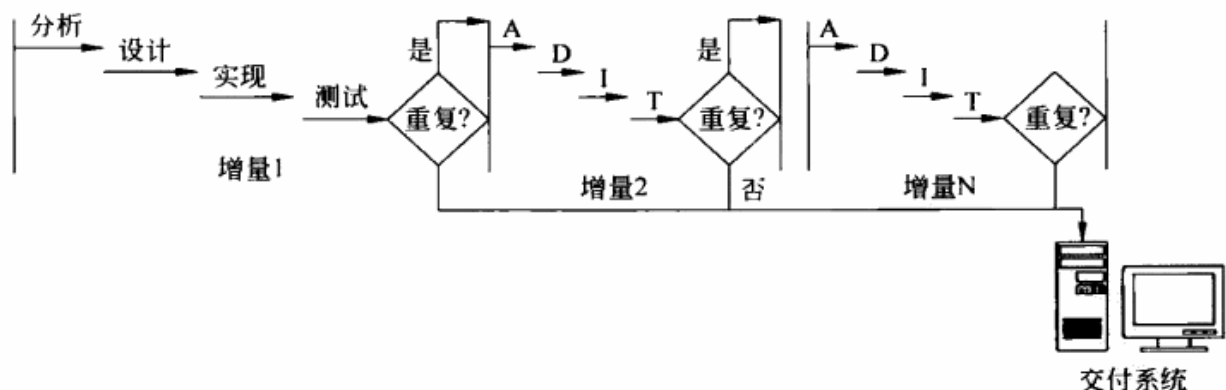


图 1-2 增量模型

增量模型有无数的例子，包括演化模型、增量模型、敏捷模型。在迭代范围大小、迭代持续时间和严格程度方面，不同的增量模型之间有很大的差异。通常情况下，迭代

模型中能够首先创建一个具备部分功能和性能，而且完全集成的可工作的系统，该系统比顺序模型生命周期中的可集成系统创建得要早。

在生命周期早期，提供可用的测试系统对测试经理而言是非常有帮助的。尽管如此，迭代模型也会给测试经理带来很多测试问题。

第一个问题是，在首个迭代之后的每个增量过程中，能够对前一个增量中的所有功能和性能进行回归测试的需求。由于最重要的功能和性能基本上是在早期的增量过程中开发的，这些功能和性能是如此重要而不能被破坏，但是，当我们对基本代码频繁进行大的变更时——每次增量过程都试图加入过多的新代码以及和前一个增量一样多的变更后的代码——回归测试的风险就会很高。为了避免这种高风险，就应该要考虑进行不同程度的自动回归测试。

第二个问题是，没有考虑到会产生的缺陷以及如何去处理和应付。这个问题意味着当业务分析人员、设计人员和程序员的工作时间被全部安排到后续增量的工作中时，测试人员还在对当前增量进行测试。也就是说，我们允许不同活动在各增量过程之间重叠，而不是要求每个增量在下个增量开始前必须完全结束。初看时这样做似乎很有效率。但一旦测试团队开始定位缺陷，和该缺陷相关的业务分析人员、设计人员和程序员就会面临超工作量的负荷。

最后一个问题，这个问题在敏捷开发中特别普遍，即测试缺乏严密性以及不被重视。但是也有例外——RBCS 的客户借鉴我们在测试过程中的丰富的经验，成功运用了敏捷开发模式（Scrum 模式），这些顾客找到了一些方法将正式的测试集成到敏捷软件开发生命周期中。但是，敏捷开发社区的最新观点已经不再考虑规范化测试和规范化测试所需的关键设施。例如：一个敏捷方法的支持者甚至批评规范化的缺陷跟踪系统，而该系统实际上是现实项目中用得最多的一种工具。

不过，这些问题都是可以避免的，但还是需要测试经理协同项目管理团队对这些问题进行细致管理。我们会在本书后续章节介绍解决这些问题的办法。

## 1.5 螺旋生命周期模型

最后，我们来看一下螺旋模型，如图 1-3 所示。该模型使用一个早期原型来设计系统，通过一系列对原型的测试，然后对其进行重新设计和构建，直到所有有风险的设计决策通过测试验证有效（或者验证无效而被拒绝）。该方法的实例就是螺旋模型，由 Victor Basili 首先提出。

该模型一般应用于超大型且复杂项目，特别是像仍旧在开发中的美国导弹防御系统。我曾经使用螺旋模型以及一些新技术来开发电子学习这类的小项目。如果您的项目有很多的未知因素，那么螺旋模型通常是非常有用的。

尽管如此，测试经理还是要负责处理螺旋模型带来的测试问题。第一个问题是由该模型的特性所决定的，即该系统的设计会经常改变。在项目的早期，对于测试用例、测试数据、测试工具和测试环境而言，它们的灵活性非常重要。如果测试经理过分相信生



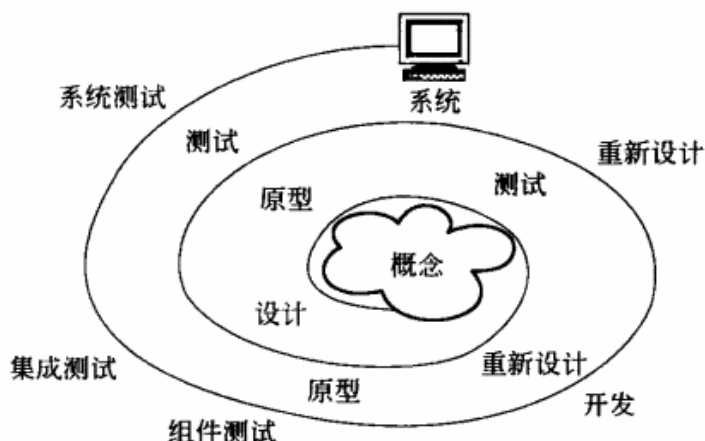


图 1-3 螺旋模型

成测试数据的某种特殊方法，那么，当系统数据目录结构有重大改变时，比如说从关系数据库转变为 XML 文件，就会出现严重的测试工作产品的返工问题。

第二个问题是由早期测试模型的独特性和实验性引起的。早期原型的测试是用来找出我们未知的问题，对有疑问的设计进行严格的测试以找出不能工作的部分。显然，对产品质量建立信心很典型不会是我们此时的目标。早期测试的不同测试目标，这些目标在最终阶段演变成更典型的测试功能，都要求测试经理去按照项目进度改变计划和策略。也就是说，灵活性是重中之重。

最后的问题是，如果我们用螺旋模型尝试处理重复的原型构建过程中的未知因素，那么项目进度会变得不可预测。进行测试估算和计划会遇到麻烦，如果同时要处理其他项目，就更加困难。

当然，这些问题都可以解决，但是如果测试经理不能正确处理的话，还是会很麻烦。

## 1.6 测试级别

基础级大纲引用了以下 4 种典型的测试级别：

- 单元或者组件。
- 集成。
- 系统。
- 验收。

基础级大纲提到了这些级别中存在不同变种的原因，特别是针对集成测试和验收测试。

集成测试可以是组件集成测试，也就是将一系列组件集成为一个系统，并对该过程进行测试。或者也可以是系统集成测试，也就是将一系列系统集成成为一个综合系统，对该综合系统从开始到多个系统乃至所有系统的集成过程进行测试。

### ISTQB 词汇表

**综合系统：**嵌入在网络不同级别和不同域中多种多样的分布式系统，它们彼此关联，并且在较大范围内有共同的问题和目标。

验收测试的种类，正如我们在基础级大纲中讨论过的，包括用户验收测试以及规范验收测试。

除了这4个级别及它们的不同变体，在高级大纲中，我们应该记住某些项目也许还需要一些额外的测试级别，这些级别包括：

- 硬件—软件集成测试。
- 特性交互测试。
- 客户产品集成测试。

对于软件测试的每个级别，我们还应该找到以下大多数内容：

- 清楚定义的测试目标和范围。
- 可追溯的测试依据（假如可用）。
- 确定各测试级别和系统生命周期的入口和出口准则。
- 测试交付物，包括预期的结果报告。
- 采用的适合于不同级别、团队和系统内固有风险的测试技术。
- 度量和度量元。
- 适用于各级别的测试工具。
- 如果需要，和组织标准或者其他标准保持一致。

当RBCS对测试团队进行评估时，我们发现有些团队在测试中使用了测试级别方法但是却将这些级别分隔后进行操作——这样做通常会导致效率低下并引起混乱。我们会在第3章详细讨论这个问题，并通过使用测试方针文档，紧密地协调和沟通好测试相关人员，协调不同测试级别，避免测试工作中的重复和漏洞，从而减少因结果不一致而产生的困惑。

让我们一起来仔细看一下这个如何配合的概念。作为例子，我们将使用图1-1中的V模型，并且假设以下谈论有关系统测试级别。

在V模型中，通过一组相互配合的测试过程，测试计划和项目计划会同时进行。也就是说，测试从项目一开始就介入了。

一旦测试计划通过之后，就开始进行测试控制。测试控制会从整个测试的开始一直到测试结束。测试的分析和设计、代码的实现和执行、出口准则的衡量，以及测试结果报告都会根据该计划进行。任何偏离该计划的行为都会被合理控制和管理。

测试分析在测试计划之后立即开始，有时同时进行，而测试的分析设计与需求确定、概要设计、详细设计是同时进行的。测试实现包括测试环境的实现，在系统设计的时候开始进行，在测试执行开始之前完成。

当测试条件符合测试入口准则时，开始进行测试执行。更确切点说，当大多数的条件符合入口准则，而其他不确定的入口准则被舍弃的情况下，测试执行开始进行。依照V模型理论，入口准则包括成功完成组件测试和集成测试级别。测试执行会一直进行直到测试条件符合出口准则，尽管某些测试经常会舍弃某些出口准则。

在测试执行的过程中会衡量测试的出口准则以及汇报测试结果，在测试执行过程完成之后会接着执行测试结束活动。

测试活动彼此之间，测试活动和剩余系统生命周期之间是不会自动进行准确配合的，



如果没有事先周到的计划，这一过程也不会在整个测试过程中持久进行。

对每个测试级别而言，不管选择哪种软件生命周期或者测试过程，测试经理必须负责进行相互配合的工作。这种配合不仅必须在测试和项目的计划阶段进行，同时也要在测试的控制阶段进行以保证持续的配合。

不管选择哪种测试过程或者生命周期，每个项目都有自己的独特选择。这在一些复杂项目中尤其明显，比如有军事背景的一些综合系统项目以及 RBCS 的一些大客户。这种情况下，测试经理不仅需要确保不同测试，还需要对这些测试过程进行修改。现有过程的模型，不管是测试本身或者整个软件生命周期，对于复杂项目的支持都不够好。

### 1.6.1 两个将测试集成到生命周期的案例研究

ISTQB 中两个重要概念是将测试集成到生命周期以及将测试作为一种预防性活动。我们通过一个网络应用的项目来看一下这两个重要概念的运用。在这个项目里，性能和扩展性是潜在的主要风险。如果不对服务器进行扩展而直接用来处理成千上万网络应用，肯定会发生很多金融灾难，顾客满意度也会急剧下降。

所以，在这个项目上，我们对交付的系统采用 4 种主要活动来确保它的性能和扩展性：

- 设计阶段，我们会按照推荐设计做静态性能测试，这通常是对性能和负载模型的电子表格进行审核和复查。
- 当以上这些测试结束之后，由于还是在设计阶段，我们接着使用商业模拟软件对推荐的设计进行静态性能分析。
- 随着编程的深入，程序员在编码实现了设计要求后进行单元性能测试，单元性能测试的这些方法也被用于构建系统性能测试环境。
- 最后，当该系统各种特性已完备，我们开始进行系统测试（假设该项目遵循软件生命周期的正常进程）。这个时候，我们对已经实现的设计进行系统性能测试。

以上方法将系统测试中发现的严重性能问题减少到可控制的数量。那不是说我们没有发现性能缺陷——我们发现了不少，但是它们可以很快地解决，而且不需要对整个设计做大改动，或者影响项目进度。

让我们再来看一个开发和测试任务在生命周期中进行交互的例子。这个例子是从一个大型的复杂项目而来，该项目用来开发一个北美地区娱乐网络的综合系统。如图 1-4 所示为系统集成阶段的入口准则，这也是系统交付运营前最后一个正式测试阶段。

- 入口准则 1 要求之前的集成测试阶段顺利退出。这是一个非正式的测试阶段，它基于风险驱动基准进行设计，用来检查不同系统接口是否正常工作。
- 入口准则 2 要求我们提供审核通过的主要规格说明文档的最终版本，这里我们嵌入了一个生命周期的假设，也就是我们在遵循这个生命周期生成这些文档。
- 入口准则 3 要求系统集成测试计划和系统集成测试审核通过。开发和项目管理团队需要评审该文档并进行批准，只有这样才能将开发和测试行为集成起来。

1. 之前的集成测试已经退出。
2. 主要规格说明文档的最终审核通过的版本存在并已提供给负责系统集成测试的相关人员。
3. 该文档（系统集成测试计划）和系统集成测试套件文档都是最终审核通过的版本。
4. 网络团队负责配置用于测试的整个系统。而且所有的集成组件，包括硬件和软件都必须是正确的版本。
5. 开发团队提供有版本控制的完备系统用于集成测试，这些系统至少需要完成一个周期的合理的系统测试。
6. 开发团队提供所有在入口准则第5条中提到的软件所必需的文档。
7. 开发团队有备好的计划用来解决所有已知“必须解决”的缺陷。

图 1-4 入口准则实例

- 入口准则4要求建立用来测试的实例系统。软件生命周期关于该系统的要求是我们有权限操作这个实例系统，而且该系统已经做好充足准备进入生产环节。很显然，这是一个很特别的标准，只有那些进行全新系统部署的工作人员才可能提出这样的条件。
- 入口准则5要求开发团队提供有版本控制的，具有正式配置管理的完全的系统。我们进一步要求这些系统至少已经完成一轮可行的系统测试。这不是要求系统测试全都通过，但是至少说明已经运行了充足的系统测试，我们对系统提供的主要功能比较满意。
- 入口准则6要求不仅获得系统，获得测试过的系统，还包括获得需要的各种文档。
- 最后，入口准则7要求制定一个计划解决所有必须解决的缺陷。这样对开发和生命周期提出了一个新的需求：控制集成测试所需的时间。

我们可以看到，这些入口准则为集成测试、预测试行为和整个开发生命周期提供了一个紧密的联系纽带。

## 1.6.2 软件生命周期中的测试练习

读一下HELLOCARMS系统需求文档，注意有些还没有完成的章节用[TBD]来标注，表示还没有最终定型。

假设这个项目会遵守迭代生命周期模式，需要经过5个迭代，每个迭代中构建的特性和功能遵循基于潜在需求的优先级。

按照ISTQB基础测试过程，列出该项目，包括每个迭代所需要的测试活动。

尽管有很多种解决方案，接下来我们仅选取一种可能的方案并进行讲解。

## 1.6.3 软件生命周期中的测试练习参考答案

首先，由于我们遵循的是ISTQB基本测试过程，让我们回顾一下关于过程的概念：

1. 测试计划和控制。
2. 测试分析和设计。
3. 测试实现和执行。



4. 评估出口准则和报告。

5. 测试结束活动。

对于 5 个部分的每个阶段都会由各自相应的活动组成。但是有些迭代可能没有包括所有的阶段。

解决办法可以按照以下结构构建：

1. 初期项目计划阶段

1.1 测试计划和控制

1.1.1 创建系统集成测试计划

1.1.2 参加总体项目计划

1.2 测试分析和设计

1.2.1 进行质量风险分析

1.2.2 列出每个迭代所需的测试套件

2. 第一次迭代（很高优先级特性）

2.1 测试计划和控制

2.1.1 调整计划以适应本次迭代的需求

2.1.2 在迭代过程中指导测试工作

2.2 测试分析和设计

2.2.1 调整质量风险分析

2.2.2 为本次迭代设计测试套件和测试用例

2.3 测试实现和执行

2.3.1 实现本次迭代的测试套件

2.3.2 执行本次迭代的测试套件

2.4 评估出口准则和报告

2.4.1 将测试结果与测试计划中的出口准则做对比

2.4.2 将测试结果向项目管理团队汇报

3. 第二次迭代（高优先级特性）

3.1 测试计划和控制

3.1.1 调整计划以适应本次迭代的需求

3.1.2 在迭代过程中指导测试工作

3.2 测试分析和设计

3.2.1 调整质量风险分析

3.2.2 为本次迭代设计测试套件和测试用例

4. 测试实现和执行

4.1.1 实现本次迭代的测试套件

4.1.2 执行本次迭代的测试套件

5. 评估出口准则和报告

5.1.1 将测试结果与测试计划中的出口准则做对比

5.1.2 将测试结果向项目管理团队汇报

## 6. 第三次迭代（中等优先级特性）

### 6.1 测试计划和控制

#### 6.1.1 调整计划以适应本次迭代的需求

#### 6.1.2 在迭代过程中指导测试工作

### 6.2 测试分析和设计

#### 6.2.1 调整质量风险分析

#### 6.2.2 对本次迭代设计测试套件和测试用例

### 6.3 测试实现和执行

#### 6.3.1 实现本次迭代的测试套件

#### 6.3.2 执行本次迭代的测试套件

### 6.4 评估出口准则和报告

#### 6.4.1 将测试结果与测试计划中的出口准则做对比

#### 6.4.2 将测试结果向项目管理团队汇报

## 7. 第四次迭代（低优先级特性）

### 7.1 测试计划和控制

#### 7.1.1 调整计划以适应于本次迭代的要求

#### 7.1.2 在迭代过程中指导测试工作

### 7.2 测试分析和设计

#### 7.2.1 调整质量风险分析

#### 7.2.2 为本次迭代设计测试套件和测试用例

### 7.3 测试实现和执行

#### 7.3.1 实现本次迭代的测试套件

#### 7.3.2 执行本次迭代的测试套件

### 7.4 评估出口准则和报告

#### 7.4.1 将测试结果与测试计划中的出口准则做对比

#### 7.4.2 将测试结果向项目管理团队汇报

## 8. 第五次迭代（很低优先级特性）

### 8.1 测试计划和控制

#### 8.1.1 调整计划以适应本次迭代的要求

#### 8.1.2 在迭代过程中指导测试工作

### 8.2 测试分析和设计

#### 8.2.1 调整质量风险分析

#### 8.2.2 为本次迭代设计测试套件和测试用例

### 8.3 测试实现和执行

#### 8.3.1 实现本次迭代的测试套件

#### 8.3.2 执行本次迭代的测试套件

### 8.4 评估出口准则和报告

#### 8.4.1 将测试结果与测试计划中的出口准则做对比

#### 8.4.2 将测试结果向项目管理团队汇报

### 9. 后期项目阶段

#### 9.1 测试计划和控制

##### 9.1.1 将计划的变动写入文档

##### 9.1.2 参加项目回顾

#### 9.2 测试结束活动

##### 9.2.1 确定存档或交接的测试工具

##### 9.2.2 将测试环境配置写入文档

您应该在此之后加入至少一个级别的细节，显示每个迭代所需的测试环境、测试套件和测试数据。

## 1.7 特定系统

### 学习目标:

(K2) 用实例解释对综合系统进行测试的特殊之处。

(K2) 解释为什么测试安全关键系统得到的 3 个输出在证明和规范一致性方面是必需的。

综合系统是几个独立的系统为了一个共同的目标进行组合之后得到的系统。正因为它们相互独立但又要互相配合，所以它们经常缺少简单和清晰的用户操作界面、统一的数据模型以及兼容的外部接口等。

综合系统项目可能包括以下特性和风险：

- 在相当长的时间内，商业成熟软件和一定数量的客户开发软件的不断集成。
- 重要技术、生命周期和组织的复杂性和不一致性。组织和生命周期的复杂性包括保密性、公司机密和规则等问题。
- 不同的团队有不同的开发生命周期和其他过程，特别是开发过程包含了分布式工作、内包和外包等常见的情况。
- 由于各系统间的交互导致潜在的、严重的软件可靠性问题。一个本身有缺陷的系统会对整个综合系统的性能造成一系列连锁的影响。
- 系统集成测试非常重要的一点是要包括互操作性测试，同时，定义好的测试接口也是测试所必需的。

很明显，综合系统的项目比单系统项目复杂得多。这种复杂性就要求我们必须提高组织能力、技术能力、过程把握和团队合作。当项目的规模和复杂性增加时，良好的项目管理、规范的开发生命周期、开发过程、配置管理和质量保证就变得越来越重要。

让我们花点时间来看一下软件开发生命周期和综合系统项目之间的复杂关系。正如之前所提到的，对于综合系统而言，我们一般会对各级别进行集成。首先，我们会对每个系统进行组件集成，接着进行系统集成，因为我们在构建综合系统。



一般情况下，除非所有的系统恰好在同一个组织内开发，而且该组织的软件开发团队遵循相同的方法，那么我们就可以使用统一的版本管理控制系统和过程。不过事实上，当我们对大公司进行评估时，我和我的同事几乎看不到这种情况。所以，软件开发中会有很多不同种类的版本管理控制系统和过程。

通常情况下，项目都会持续很长时间，我曾经见过有些项目计划时间长达5~7年。如果一个综合项目只集成5~6个小系统就会被认为很小。如果只需要持续1年，或者只需要40~50个人的参与还是会认为相对比较小。在这样的项目中，会有不同的测试级别，通常这些级别由不同的团队负责。

如果项目比较庞大和复杂，我们可以对具体职责进行分解，从而相对容易地进行交接和转移。因此，在项目成员之间，我们需要正式的信息交换机制进行无缝沟通，这一点，在项目的关键时刻尤为重要。

尽管我们只是进行纯粹的商业成熟系统集成，这些系统，通常是客户系统，实际上是会经常演变的。我们会面临两种挑战：（1）在协调沟通单个系统的开发时所遇到的测试管理方面的挑战；（2）综合系统中不同级别的回归测试会遇到测试分析方面的挑战。

尤其对于“商用现成品系统”，有时候在没有太多征兆的时候，外部实体或者事件就会触发维护性测试，例如：系统的废弃、淘汰或者升级。

假如在综合系统中考虑基本的基础测试过程，该测试过程不是二维的，而是像金字塔一样的结构，如图1-5所示。

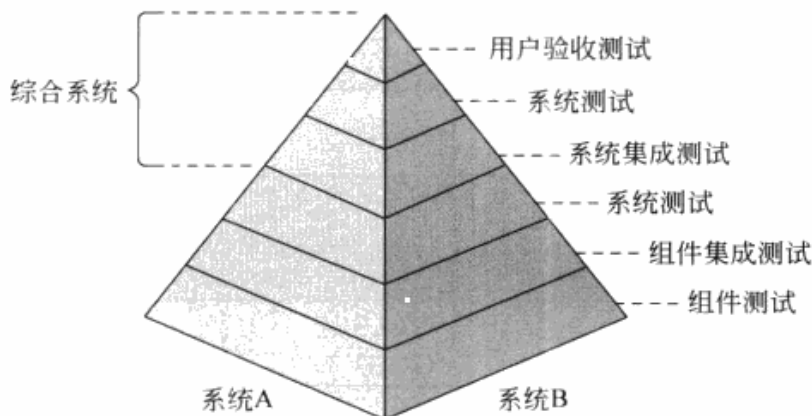


图 1-5 综合系统项目的测试级别金字塔

在金字塔的底部有组件测试，每个独立的系统都有一个对应的组件测试级别。底部往上是组件集成测试，每个独立的系统有一个对应的组件集成测试级别。接着是系统测试，每个独立的系统也有一个对应的系统测试级别。

需要注意的是，对每个级别而言，如果这些系统来自不同的卖方，我们就有不同的组织隶属关系。我们也可能有不同的团队隶属关系，因为需要不同的团队来分别负责组件测试、集成测试和系统测试。

沿着金字塔继续往上，就是系统集成测试。再往上是系统测试，它是横跨整个系统的单一测试级别，侧重于覆盖全系统的端到端测试。最后是用用户验收测试。对于这些不同的测试级别而言，就算只有一个组织来执行，我们也要分成不同的团队来负责。

### 1.7.1 综合系统项目案例研究

表 1-1 展示了一个复杂的综合系统的预集成测试的实例，这个复杂的综合系统是北美地区的一个娱乐系统，由 7 个主要系统组成，如表 1-1 所示的 A~G：

- 互动式语音应答服务器（IVR）硬件，即客户端服务器。
- 将该系统集成在一起的局域网和广域网。
- 互动式语音应答服务器应用程序，该软件运行于 IVR 服务器中用来和客户进行交流 and 互动。
- 客户服务程序，在呼叫中心运行，客服专员会对这些求助电话进行处理。
- 呼叫中心基础设施，例如电脑电话集成单元。
- 存储客户交易信息以及其他商业智能信息的数据仓库。
- 针对服务器上所有信息的内容管理软件。

表.1-1 综合系统中的迭代集成

系 统	BB0	BB1	BB2	BB3	BB4	BB5
A. IVR 硬件	Alpha		Beta	Quad		Prod
B. 网络	LAN					WAN
C. IVR 应用程序	C1-E1	C2-D1	C3-D2 C3-G3	C4-D3 C5-F1 C6-E4	C7-F2 C8-D6	C9-D8
D. 客户服务程序		D1-G2	D2-C3	D3-C4 D4-G4 D5-E5	D6-C8 D7-F3	D8-C9 D9-F5
E. 呼叫中心	E1-C1 E1-G1	E2-G2	E3-G4	E4-C6 E5-D5		
F. 数据仓库				F1-C5	F2-C7 F3-D7 F4-G5	F5-D9
G. 内容管理	G1-E1	G2-E2	G3-C3 G4-E3	G4-D4	G5-F4	

预集成测试也就是在系统集成测试之前进行的测试。在这个阶段，我们定义 6 个主要的构建过程，也叫做迭代。这些迭代在以上的 BB 表中定义为 BB0~BB5。每个迭代允许在各系统间的一到多个接口间进行测试。该接口在以上的表中用字母/数字~字母/数字的组合进行区分。字母代表系统，数字代表接口的序列号。注意到有些接口被单个系统用来和超过一个以上的其余系统进行交互。例如：在迭代 BB0 中呼叫中心的 E1 接口和 IVR 应用程序的 C1 接口以及内容管理软件的 G1 接口进行交互。

#### ISTQB 词汇表

**安全关键系统：**该系统的失败或者失灵会导致人员的死亡或严重伤害，或者对设备和环境有严重的破坏。



这个练习的目的，你们也可能猜出来了，就是要在正式的系统集成测试之前测试尽可能多的关键接口。这个早期的测试会使我们在正式的集成测试开始之前找到潜在的威胁和设计的缺陷并努力消除它们。

## 1.7.2 安全关键系统

简单地讲，安全关键系统指的是那些关系到生命的系统。安全关键系统的失败，哪怕只是暂时的性能或可靠性降级，或者是在支持工作中不期而至的副作用，都可能导致人员的伤害或死亡。

和综合系统一样，安全关键系统也有一些特定的特性和风险，包括：

- 由于缺陷可以导致人员伤亡以致引起法律处罚，所以充分的测试经常被用来作为证据以减轻缺陷引起的处罚。
- 出于很明显的理由，不同的法规和标准经常被应用于安全关键系统。法规和标准可以用来约束过程、组织结构和产品。和平常对项目的约束不同，通过采用一种特别定制的约束，而不是通过牺牲质量来保证进度、预算或功能。简而言之，对质量的重视是最重要的优先级。
- 通常来说，开发和测试都会有很严密的方法。纵观整个生命周期，从法规要求到测试结果，所有步骤都必须是可追溯的，这样才能帮助证明是符合法规和标准的。所以，不仅要求广泛而详细的文档，还要提供高度的可审核性，甚至是让非测试专家进行审查。

假如法规要求是强制的，那么审计是经常采用的一种手段。需要做到从法规的需求到开发再到测试结果的整个过程的跟踪。审计通常由一个外部组织执行。所以，要做到前台有据可循，而后台则进行严格审核活动和后台的审计/遵守（法规）活动同时从人员和过程的角度影响着管理、开发、测试以及主管权威机构。

早在生命周期的设计阶段，项目团队就使用了安全分析技术来确定潜在的问题。单个失效常常可以通过系统的备份来解决。

在某些情况下，安全关键系统可以看做是复杂系统甚至是综合系统。在另外一些情况下，有些非安全关键的组件或系统会被集成到安全关键系统或者综合系统中。例如，网络或者通信设备不是天生的安全关键系统，但是如果它们被集成到一个紧急调遣或者军事系统中，就会变成安全关键系统的一部分。

在这种情况下，正式的质量风险管理是非常必要的。幸运的是，我们已经有了很多类似的技术，例如，失效模式和效果分析；失效模式、效果和关键分析；机会分析以及软件故障常见原因分析。我们来看一下下面这个例子：

RBCS 的一个客户是一家建造规范化医疗设备的公司，它们遵守美国食品药品管理条例并以此管理其生产的产品。针对该测试的规则包括以下几点：

- 每一个测试用例都应该可以追溯到需求规格说明。测试用例就是该组织用来证明产品满足需求的一种方法。
- 需要某个审计机构定期对测试进行审核以确保和条例的一致性。审计过程包括对测试本身以及测试结果的审计。

- 为了支持该审计，该机构不仅仅要存储测试也要存储测试结果作为证据，包括屏幕截图等。

#### ISTQB 词汇表

**度量元：**一种用来度量的度量标准和方法。

**度量标准：**约束数据分析类型的标准。

**度量：**给实体赋予一个数值或类别以描述其某个属性的过程。

**测量：**度量时赋予实体某个属性一个数值或一个类别。

由于以上规则相当的正式和官方，所以该组织灵活地使用了符合该规则的最轻量化的文档。这个组织遵循 Scrum 生命周期，即一种敏捷开发生命周期模型。但是该组织已经对软件生命周期进行了修改，用以生成以上规则所需的文档。

## 1.8 度量元和度量

### 学习目标：

- (K2) 描述和比较标准测试相关的度量元。
- (K3) 通过衡量测试对象和测试过程来监视测试行为。

在本书中，我们使用度量元和度量来建立我们的期望目标并通过该目标来指导测试。在整个软件开发生命周期中我们应该应用度量元和度量。因为良好的度量元和度量加上恰当的项目目标，能够使测试分析师用一致连贯的方法追踪测试和质量结果并汇报至管理层。

缺乏度量元和度量会导致过分主观的评定测试质量，而且在软件生命周期的最后阶段容易对测试结果的意义产生争议。同时也会导致对测试效率、测试有效性和价值等缺乏清晰的沟通理解。

我们不仅需要度量元和度量，还需要基线。一个好的度量元结果应该是怎样的？一个可接受的结果？一个不可接受的结果？如果没有预先定义的基线，就不可能有成功的测试。事实上，当我和同事对我们的客户进行评估时，我们经常发现测试团队在有效性和效率上错误地定义度量元，因为没有基线，就会出现差的、不现实的期望值（期望当然也就无法实现）。

对于什么可以作为度量元以及如何在度量过程中进行跟踪，答案有很多。考虑以下几点：

- 计划的进度和覆盖率。
- 测试所需的需求、进度、资源和任务。
- 工作量和资源的使用。
- 测试的里程碑和范围。
- 计划和实际的成本。



- 风险，包括质量风险和项目风险。
- 缺陷，包括总共发现的、总共修复的、现存未解决的、平均解决时间、配置、子系统、优先级，或者严重性分布。

在测试计划阶段会建立期望结果，也就是之前提到的基线。作为测试控制的一部分，我们会将测量的实际结果及趋势和这些期望结果进行比较，找出两者的差别，然后按照指示对我们的方法进行调整。作为测试汇报的一部分，我们可以用大家一致认可的客观的、实际的、可实行的度量元向管理层不断地解释过程、产品和项目的各个方面。

我们来看一个例子，这个项目在本书中被使用了很多次，图 1-6 展示了一个综合系统项目的总计发现缺陷数量和缺陷解决数量。可以看到，这个项目真的是有很多问题和麻烦。总计缺陷数量非常多，巨大的未完成缺陷表，而且周缺陷新发现比率一直很高。好消息也很明显，那就是周缺陷修正率和缺陷发现率几乎一样。虽然缺陷发现率没有下降，但是至少未完成缺陷率也没有增加。

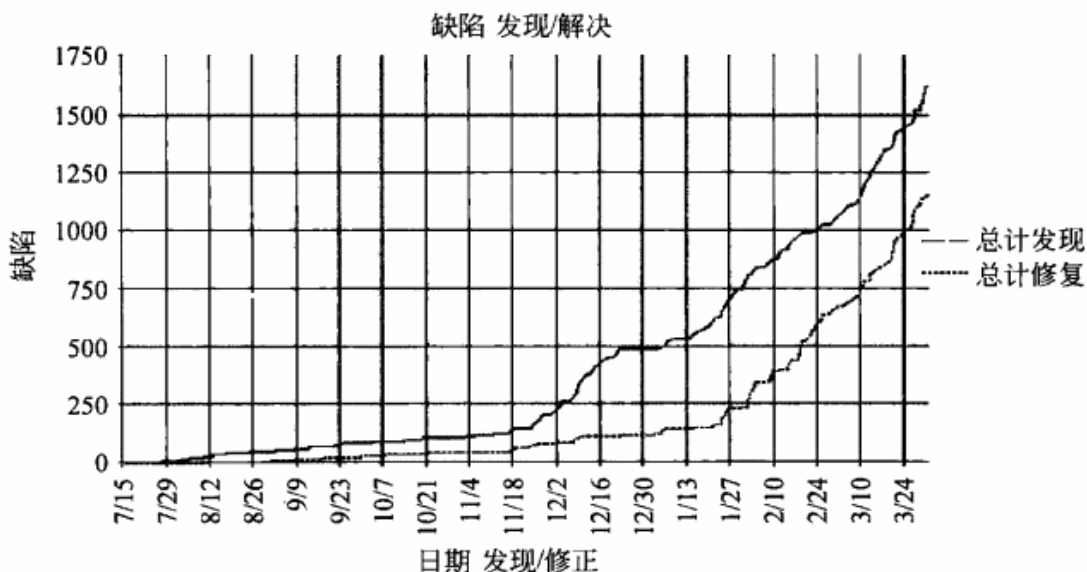


图 1-6 缺陷度量元

编制一个测试度量元和度量，需要考虑 3 个主要的方面：定义、跟踪和汇报，让我们先从定义讲起。

编制一个成功的测试度量元，需要定义一组有用的、合适又精确的针对质量和测试的度量元。要避免使用太多的度量元，太多的度量元会导致测量非常困难，代价也极其昂贵，同时会让旁观者和投资者感到困惑而不是振奋。

另外，还要保证大家对这些度量元都有一个统一的理解，这样才可以减少在测试结果、分析和趋势的衡量中出现争议和分歧。如果对某个特殊的度量元的意义，每个人都有不同的意见，那就绝对不要采用这个度量元。

最后，定义度量元的标准既要按照过程或者任务的实现目标，也要考虑不同的个人、团队、组件、系统的特性。

Victor Basili 先生著名的目标问题度量元技术介绍了一种生成有效度量元的方法。该技术中，我们从努力要实现的目标出发，这里特指软件测试要实现的目标，到我们要回答的问题，来确认是否最终达到了具体度量元的目标。



例如,测试的一个典型目标是建立信心。那么一个很自然的问题就是“这个系统测试到什么程度了?”覆盖率的度量元包括测试覆盖需求的百分比、测试覆盖的分支和语句,测试覆盖的接口等。

让我们继续谈跟踪。在一个度量元项目中跟踪是反复发生的活动,自动化工具的使用可以减少捕获、跟踪、分析、汇报和测量该度量元需要的时间。

一定要对具体的度量元进行主观和客观的分析,尤其是当有趋势表明这个度量元可以有多种不同意义的解释时。尽量避免太快下结论,应该鼓励其他人一起参与进来并达成统一观点之后再发布。

通常情况下,由于人们对某种趋势的兴趣会影响对某些特定度量元和度量的理解,因此要时刻注意这种趋势并对此进行有效管理。每个人都认为自己是客观的,但是人们的兴趣对他们的结论会产生很大的影响。

最后,我们来看一下汇报。最重要的一点是,汇报度量元和度量要对管理层和投资者非常有用,而不是混淆或者误导他们。通常来讲,这就需要合理定义的度量元和细心的跟踪。但是有时候尽管度量元非常有意义且又很清晰,如果表述不清楚,也会让大家很困惑。Edward Tufte 的系列书籍是个宝藏,里面提到很多关于如何撰写好图表的知识,对于汇报很有用<sup>1</sup>。

Tufte 肯定会认为基于度量元的一个好的测试报告应该容易理解,不会过于复杂,当然也绝不会引起歧义。它们也应该将观众的焦点聚集在重要的事项上,而不是一些琐碎的小事。通过这种方法,基于度量元和度量的好的测试报告会帮助管理层更好地指导项目并最终获得成功。

不是所有度量元的图形显示都一样,或者有一样的用处。某一时刻的一个数据快照,正如表中所示,可能很好地表现了某些信息,比如说在某些关键质量风险域中计划达到和实际达到的覆盖率。一个趋势图可能会很好地表现出别的信息,比如从测试开始的总计缺陷数量和总计解决缺陷数量。一个原因或者关系分析图,对表达一些别的信息非常有用,比如一个扩散图表明了测试者经验年限和缺陷报告被拒绝的关系。

### 1.8.1 度量元和度量的练习

假设您对 HELLOCARMS 项目使用基于风险的测试策略。请说出一到多种您可以用来在测试执行时报告剩余质量风险级别的度量元。我们在下面提供了一些可能的答案。

### 1.8.2 度量元和度量的练习参考答案

还有很多种测试执行时汇报剩余质量风险级别的不同方法:

- 一张列出主要风险条目的表,列出每个风险条目相关的计划测试用例数量、成功测试用例数量和失败测试用例数量。
- 一张和上述相同的表,不过加上两组数据,每个风险条目中报告的缺陷数量和被

---

1 Tufte 的系列丛书包括定量信息的图形显示、预想信息和可视说明。

解决缺陷的数量。

- 一个柱状图用来表示测试用例对计划和实际的风险条目的覆盖率。
- 一个相同的柱状图，加上一组柱子用来表示每个风险条目上发现的缺陷百分比。

注意您需要从风险条目到测试用例以及/或者缺陷报告这些度量元的可追溯性来进行工作。

#### ISTQB 词汇表

职业道德：未定义。

## 1.9 职业道德

### 学习目标：

能回想起该部分内容即可。

很多专业人士有自己的职业道德标准。在专业领域里，所谓道德就是公认的行为准则，包括某一类特殊行为，特别的团队和文化等<sup>2</sup>。

因为测试人员经常有机会接触绝密的信息和资料，职业道德准则可以帮助他们适当地使用那些信息。而且，基于所给条件和限制，测试人员要运用职业道德准则选择可行性行为来应对特定的情况。“最佳可能”是对所有人的，不只是对测试人员。

和您分享一件关于职业道德的事情。我是 3 个跨国软件测试顾问机构的管理人员，它们是 RBCS、RBCS NZ 和 PureTesting。同时我也为 ISTQB 和 ASTQB 董事会服务。正因为如此，我可以看到软件测试顾问领域的其他竞争者看不到的某些项目的未来发展方向。

在某些情况下，比如帮助撰写大纲，这是为了向大家说清楚其中的商业利益。我会帮忙编撰基础级大纲和高级大纲。

在其他情况下，比如制定认证考试内容。我同意 ASTQB 同事的观点，那就是我不应该参加认证考试内容的制定。直接加入认证考试系统会让我可能有意或者无意地把我的培训材料变成“指导认证考试”。

当您在以后的职业生涯中成为了一名测试人员，就会有越来越多的机会显现您的职业道德准则，或者由于道德缺乏而背离该准则。大力提倡测试人员强烈的职业道德永远不会太早。

在高等测试级别中，ISTQB 组织希望证书持有者坚持以下道德准则：

- 公开——软件测试认证人员应该做和公众利益一致的事情。比如，您在做一个安全关键系统时，有人要求您偷偷取消某些缺陷的报告，这就是个职业道德问题。
- 顾客和雇主——软件测试认证人员的行为应该符合其顾客和雇主的最佳利益，当

---

2 摘录自 dictionary.com。



然也要和公众利益相吻合。比如说,如果您知道雇主的主要项目有问题,然后卖空该股票并将该信息泄露到网上,那绝对是道德缺失,还有可能构成犯罪。

- 产品——软件测试认证人员应该尽可能保证他们所交付的产品(包括对产品和系统的测试)符合最高的专业标准。比如说,为了使客户在下一个项目中继续雇佣您,作为顾问的您却故意遗漏测试计划的一个重要细节。这也是职业道德的缺失。
- 判断——软件测试认证人员应当在他们的专业判断中保持诚信和独立。例如,如果一个项目经理要求您不要汇报某些部分的缺陷,因为这样可能会引起商业赞助商潜在的不利反响。这种情况是对您的独立性的考验,如果您按照项目经理的要求做了,就是您没有遵守职业道德。
- 管理——软件测试认证经理和主管应当在软件测试管理中促进并遵守职业道德方法。比如说,您欣赏某一位测试人员而不喜欢其他人,而原因却是您希望和他姐姐建立一种暧昧的关系,这是一种严重失败的管理职业道德。
- 专业——软件测试认证人员应当树立专业的诚信和威望,使其和公众利益保持一致。例如,如果您有机会向您孩子的同学或者您配偶的同事介绍您的职业,您要为您所从事的职业感到自豪并解释测试对社会起到的积极作用。
- 同事——软件测试认证人员应该公平对待和支持他们的同事,并促进和软件开发人员的合作。例如,通过操纵测试结果导致一个您讨厌的程序员被辞退的行为是极其不符合职业道德的。
- 自我——认证软件程序员应当在他们的专业锻炼中不断地参与职业生涯的学习,并通过职业练习提高他们的职业道德方法。例如,参加课程、阅读书籍、在会议上讲解您的职业。这些都可以提高自身专业技能。参加这种活动对于自身都是非常有帮助的,而且会提高职业道德。

## 1.10 认证考试模拟题

在每个章节的最后,我们会准备一个或多个认证考试模拟题来加强您对该章内容的理解并用来准备将来的 ISTQB 高级测试经理模块的认证考试。

1. 以下哪几项可以作为项目需求规范说明阶段测试的具体例子?
  - A 需求回顾会议
  - B 业务分析师引出需求
  - C 数据库管理员设计表
  - D 展示需求覆盖率的测试结果报告
2. 假设您是一名项目测试经理,这个项目要创建一个家庭使用的可编程恒温器来控制中央供热系统、空气流通设备和空调系统。这个项目遵循顺序生命周期模型,具体地说是 V 模型。现在,系统架构师按照之前发布的需求规格说明发行了第一版设计说明草图。识别以下所有目前需要执行的测试任务:
  - A 从需求规范说明中设计测试
  - B 分析与设计相关的风险

- C 执行单元测试用例
  - D 撰写测试概要报告
  - E 从设计规范说明中设计测试
3. 您是一名银行质量评估小组的经理，负责银行程序的独立测试。您要负责的项目是将3种成熟系统组合成一个集成系统，并使用该集成系统管理银行的应收款项系统。识别出以下所有您将要直接管理的测试级别：
- A 对每个系统的组件测试
  - B 对每个系统的组件集成测试
  - C 对每个系统的系统测试
  - D 对每个系统的合同验收测试
  - E 系统集成测试
4. 通常情况下，以下哪几项符合安全关键系统规则？
- A 测试可追踪性
  - B ISO 61508
  - C 可用性测试
  - D 雇员评估
5. 下面哪几项准确描述了测试执行中总计发现及解决缺陷数量趋势图和总计通过及失败测试用例数量趋势图的区别？
- A 缺陷趋势图可以揭示测试过程问题而测试用例趋势图不可以
  - B 缺陷趋势图可以揭示测试覆盖率而测试用例趋势图不可以
  - C 测试用例趋势图可以揭示测试过策划问题而缺陷趋势图不可以
  - D 测试用例趋势图可以揭示测试覆盖率而缺陷趋势图不可以
6. 考虑在基础级大纲中定义的测试目标，在达到这些目标的时候，下面哪个度量元可以用来衡量测试过程的有效性？
- A 从缺陷发现到解决的平均天数
  - B 每个开发人员每天写代码的行数
  - C 回归测试在测试中所占百分比
  - D 需求覆盖率所占百分比



## 第2章

# 测试过程

教官：阿甘！你为什么这么快就将枪装好了，阿甘？

阿甘：你叫我做的，教官？

教官：【惊讶的表情，看秒表。】这是个新的纪录，如果你不去候补军官学校就读的话，将会是一个极大的浪费，有朝一日你将成为将军。阿甘，我现在命令你拆卸后重新装枪！

——在由文森格汝姆小说改编的电影《阿甘正传》中，埃里克罗斯扮演的阿甘显示了精确快速执行流程的能力。

高级大纲第2章主要描述测试过程以及在该过程中发生的活动。它为大纲后续内容建立了一个框架，通过该框架可以形象地对其余概念进行合理的组织。该章节包括7个部分：

1. 概述。
2. 测试过程模型。
3. 测试计划和控制。
4. 测试分析和设计。
5. 测试实现和执行。
6. 评估出口准则和报告。
7. 测试结束活动。

让我们依次来看一下每个部分的内容，以及它们和测试管理的联系。

## 2.1 概 述

### 学习目标：

能回想起该部分内容即可。

ISTQB 基础级大纲介绍了 ISTQB 的基本测试过程。它提供了和可定义的过程，如图 2-1 所示。这个过程由以下活动构成：

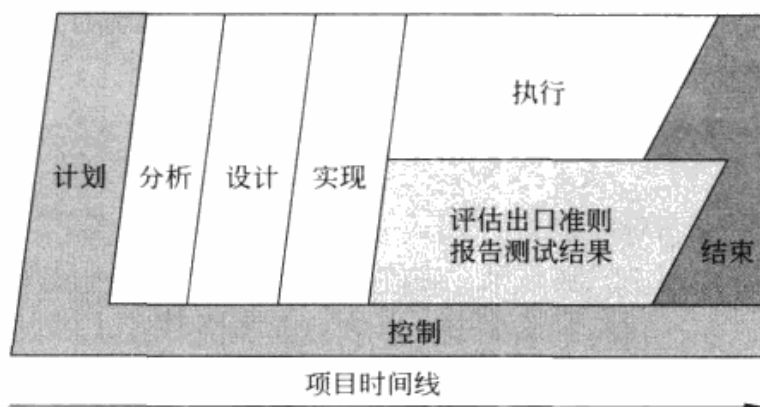


图 2-1 ISTQB 基本测试过程

- 计划和控制。
- 分析和设计。
- 实现和执行。
- 评估出口准则和报告。
- 测试结束活动。

作为测试经理，我们不会在分析和设计上花费太多时间，但是我们将关注其他活动，特别是计划和控制。

## 2.2 测试过程模型

### 学习目标:

能回想起该部分内容即可。

我们会在第 8 章对本章内容进行更详细的阐述。因此，本章没有针对测试经理的学习目标。如果您在为 ISTQB 高级测试经理认证考试学习准备，请记住 ISTQB 高级大纲覆盖了所有的 K1 级别而且是作为考点的。因此，阅读高级大纲第 2 章的这个部分只是用来提醒您记住这点。

## 2.3 测试计划和控制

### 学习目标:

- (K2) 用实例描述测试策略如何影响测试计划。
- (K2) 比较各测试工作产品并用实例解释开发和测试工作产品的关系。
- (K2) 将测试相关控制行为进行分类并以此决定测试任务的策略和目标是否达标。

对于分析式的测试过程，比如 ISTQB 定义的基本测试过程中，测试计划一般在项目开始的时候进行。理想情况下，它和整个项目的计划应该同步进行并描述测试策略如何实现。例如，在基于风险分析的测试策略中，质量风险分析会决定测试的优先级并分配



工作量。紧接着测试计划根据该优先级和计划工作量描述要执行的测试工作。我们会在第3章详细讲述风险分析、基于风险的测试和测试计划文档。

### ISTQB 词汇表

**测试策划：**制定或者更新测试计划。

**测试计划：**用来描述执行测试活动的范围、方法、资源和进度的文档。该文档定义和鉴别了相应测试项、执行的测试任务、测试者独立程度、测试环境、所使用测试设计技术和入口出口准则以及选择以上条件的原因和处理风险所需的应急计划。它是测试计划过程的文档记录。

测试策划不仅仅局限于单个级别的测试，而是需要覆盖所有发生的测试级别。也就是说，需要有一个主测试计划，横跨各个不同的测试级别，保证在合适的时间执行合适的测试。例如，如果我们认为一个质量风险域非常危险，那么主测试计划应该在生命周期的早期包含对该区域的测试，甚至在生命周期的不同级别也要包含该区域的测试。

在测试策划的过程中，我们会建立一个框架，通过了解该框架的测试依据获得测试用例、测试条件和测试过程。测试依据包括需求规范说明、设计规范说明、质量风险和其他项。如果一个测试计划得很好并且被较好地贯彻执行，那么会保证测试执行过程中顺利地执行计划执行的测试项。这个听起来简单，其实并不容易，因为该计划要处理测试依据和测试用例、测试条件和测试过程之间的复杂关系。由于测试依据存在于项目的测试策划阶段，而测试用例、测试条件和测试规程是在测试的执行阶段。

一旦我们建立了测试计划，接着就进行测试控制。测试控制其实是个持续的行为。在测试控制中，我们将实际过程和测试计划进行比较。我们会汇报测试计划执行的状态，包括可能引起的任何偏离。由于项目不是静态的，而是在不断进化的，因此测试控制在指导测试满足测试使命、目标和策略方面的作用较大。

测试控制必须对测试目标、使命和策略，或者项目变化实行相应变更。例如，我们可能发现某个区域的风险分析按照缺陷可能性来看是错误的，这样会导致重新分配剩余测试工作量和确定优先级。或者，我们必须处理测试时间被压缩的情况，运用风险分析可以指导我们合理地减少测试用例执行总量。

### ISTQB 词汇表

**测试用例：**为某个特殊目标或者测试条件而编制的一组测试输入、执行的前置条件以及预期结果和执行的后置条件，用来测试或核实程序路径是否满足特定需求。

**测试条件：**组件或系统中能被一个或多个测试用例验证的条目或事件。例如，功能、事务、特性、质量属性或者结构化元素。

**测试执行：**在组件或者系统上进行测试并产生实际结果的过程。

**测试规程：**规定了执行测试一系列行为的文档，也称为测试脚本或手工测试脚本。



**测试控制：**当监测到与预期情况出现偏差时，制定和应用一组修正动作以保持测试项目正常进行的一种测试管理工作。

测试策划和控制经常需要设计、创建或者更新文档。我们会在第3章讲述关于测试计划和控制的测试文档。

### 2.3.1 测试计划和控制的案例研究

我们用一个例子来讲解测试计划和测试控制的概念。当 RBCS 公司需要进行网站重设计时，我们雇佣了一个市场公司来帮助我们创建主要内容以及雇佣了一个网络开发公司来开发实现这个网站。我们和该网络开发公司定下工作说明书，包括合同。该工作说明书包括了我们网站的需求。

作为这个项目的一部分，我们计划去执行验收测试。验收测试的主要目的，其实和在这个级别的大多数测试一样，是要核查该次发布是否可行。由于是按照需求来制定和网络开发公司的合同的，因此我们决定使用基于需求的测试策略。

基于该策略，我们撰写一个计划覆盖所有已经陈述和隐含的需求。该计划包含了文档化的测试用例，尽管它们是逻辑级别的文档，而不是具体的文档。当然，在某些需求模糊不清的区域，我们也使用了探索性测试。

从图 2-2 中可以看到需求规格说明的两段条件如何变成了 4 个测试用例，我们可以看到前两个测试用例相对清楚。这些测试用例有很多测试条件组合的可能，这和我们已经测试过的用例非常不一样。但是，测试的准则，也就是如何确定期望行为是很清晰的：如果一个旧客户在退出登录或者根本就没有登录的时候重输订单或者送货信息，就是一个缺陷。

#### 需求概要：

当顾客下单时，如果他是一个老顾客，那就可以使用登录功能页面避免重复输入之前登录时已经输入过的所有的配送和付款信息。登录功能和顾客账户管理功能在结构和功能性方面与已经成为大多数主要电子商务网站的标准相似。

如果需要，供应商会提供符合 PCI 标准的信息并将其交由后续支付设备或者信用卡处理。

#### 测试套件概要：

1. 老顾客可以使用登录界面避免重复输入配送信息。
2. 老顾客在确定订单的时候可以避免重复输入他们的支付信息。
3. 登录和顾客账户管理功能和主流电子商务网站相似。
4. 供应商提供信息证明其符合 PCI 规范。

图 2-2 将需求转换成测试用例

第三个测试用例相对来说就比较混乱。哪个是我们期望的行为？现实中有成千上万个电子商务网站。哪一个才定义了标准？实际上，我们用 Amazon.com 作为我们的参考网站。其实不管选择哪个网站，只要该网站是主流网站就可以。

第四个测试用例从表面上看还很清晰。这里提到了符合证书，这看起来很专业，其实它也非常不清晰。如果您问“好吧，符合 PCI 需要什么条件？”也就是问，这个说明



书的内容应该是什么？鉴于供应商也是这个网站的主人——也就是说他们如果不遵守的话也会有很大的风险——所以我们决定让该公司相关人员去担心这些细节。

在表 2-1 中，可以看到测试控制机制也作为这个测试的一部分。通常讲，简单测试项目会利用简单测试控制。在本书的后续章节，会介绍复杂的测试状态报告和控制机制的例子。但是，在这个简单的项目里，对测试用例状态和缺陷状态进行追踪已经足够。

表 2-1 基于缺陷状态的测试控制

第二轮测试后的缺陷状态	
验证合格	47
待验证	1
待 RBCS 批准	5
缓期	21
验证失败	3
非缺陷	8
新缺陷	18
总计	103

表 2-1 表明通过第二轮测试之后的缺陷状态。我们先运行第一轮测试：把每个测试用例运行一次，然后找出缺陷。然后我们让软件公司修复这些缺陷之后再运行一次测试。最终，总共运行了 3 轮测试以完全满足我们的需求目标。

尽管如此，验收测试最后还是因为达不到我们的要求而失败了。一个典型的验收测试应该不是用来检测缺陷的。当然，在这种情况下，由于我们对软件公司去执行我们所提供的测试并对结果非常失望，我们只能找出缺陷。在这种情况下验收测试的失败相当于让客户来承担损失。当然我们会在之后的章节中讲到质量和测试管理相关的成本问题，但是在这里，我只能说，我们对那个网络公司很不满意。

在结束本章之前，来看一下测试计划和控制的度量元和度量。我们用以下方法来衡量该部分测试过程完成与否：

- 风险和测试覆盖率。
- 缺陷发现率和其他缺陷信息。
- 开发测试件和执行测试用例的计划时间和实际时间。

在测试计划过程中建立的任务分解结构对测试控制非常有用，它可以帮助决定我们是否按照估算和进度进行工作。

## 2.4 测试分析和设计

### 学习目标:

能回想起该部分内容即可。

本章的内容主要针对测试分析师。本章节没有针对测试经理的学习目标。

如果您学习的目的是通过 ISTQB 高级测试经理的认证考试。记住 ISTQB 高级大纲的内容覆盖了所有 K1 级别的考点，所以也会考到。因此，建议您在认证考试前，读一下高级大纲第 2 章的该部分进行回忆和熟悉。

### ISTQB 术语

**测试实现：**开发和定义测试程序的优先级、创建测试数据，或者准备测试工具、撰写自动化的测试脚本。

## 2.5 测试实施和执行

### 学习目标：

- (K2) 解释测试执行的前置条件。
- (K2) 用例子解释在测试实现早期使用不同测试技术的优缺点。
- (K2) 阐述在测试执行的时候用户或者顾客参与其中的原因。
- (K2) 描述基于测试级别的测试日志如何改变。

在测试实现过程中，测试经理需要保证已经成功完成了在测试执行前必须要处理的剩余任务。包括组织好测试规程并且测试人员可以得到该规程，也包括决定谁执行测试，何时使用以及使用哪些测试数据、测试用例、测试环境和测试工具。只有明确了这些活动，制订的工作进度才会更加合适。当然，也别忘了检查显性的入口准则（在测试计划中定义）和隐性的入口准则（可能在项目计划或者人们脑中）。

测试实现过程中有两个与测试管理相关的挑战，首先是弄清楚获得测试数据的来源和方法，利用这些来源和方法获得数据，并保证测试人员可以得到这些数据。测试经理也许需要安排使用生产或者实时数据，这个又和数据保密性有关系。您也可能需要使用工具来生成测试数据，特别是当您执行负载测试、容量测试，或者性能测试的时候。测试分析师需要创建和使用测试数据，数据经理需要保证有可用的数据和工具。

另一个测试管理的挑战是测试环境。某些项目中，保证测试环境的可用性是很庞大和复杂的事情，而且需要非常细致周详的计划工作。有些缺陷只可能在类似于运行环境的测试环境下才能被发现，而获得这种环境通常非常昂贵而且配置和管理相当困难。测试经理不得不在测试实现的过程中着手去做这项工作。这个管理包括检查测试环境依赖性，例如：有些测试需要独占测试环境。测试分析师需要配置和验证测试环境，而测试经理需要确保测试分析师得到需要的资源。否则，测试可能会被阻碍或者得到无效的测试结果。

对测试人员来说，一个好的准则就是“首先找到最紧缺的资源。”这意味着我们必须选择正确的测试，并按正确的顺序执行。所以对测试进行优先级分配很重要。在基于风险的测试策略中，我们按照风险发生的可能性和风险的严重程度分配测试优先级。

当一个测试人员完成了测试用例和测试规程之后，有一个重要的管理问题是，“文档



详细到什么程度才是正确的？”这个答案由不同的变量决定。

有时候这个答案很容易。如果您按照标准和规则来管理项目，那么这些标准或者规则就会让您明确项目需要达到的细节程度。通常情况下，只有具备非常详细的标准或者规则，才能创建更多的测试来实现工作。

但有些时候却很难确定这个问题的答案。如果对详细程度没有任何规则，那么您必须明白他们到底需要什么。测试人员的技能水平如何？测试结果的可重现性有多重要？测试规程是否应该实现自动化？是现在还是最后？

这个关于详细程度的问题是测试管理的重要问题，但是很少有测试经理能有效地进行管理。对于这种问题我会建议您评估测试用例现行的详细程度然后采取措施进行控制。

说到自动化，测试实现包括撰写自动化脚本。自动的脚本测试必须非常详细，因为它们需要告诉测试工具具体要做的事项。自动化测试脚本的合理实现对自动化投资是否有好回报具有关键作用。

### ISTQB 词汇表

**测试脚本：**通常用来指代一个测试规程的规范说明，特别是一个自动化测试规程。

测试项目中的测试实现应该在早期的什么阶段开始？这是另一个复杂的问题，特别是当项目需要详细的测试时。一方面，测试规程越详细，就需要更多的精力来实现测试，这也迫使您更早地开始。另一方面，测试规程越详细，对最后的用户界面及需求的变化越敏感，这样就迫使您晚些才能开始。只有当您用逻辑测试用例而不是具体测试用例，它会允许您早点进行测试执行，然后用一个适当的节奏继续进行。

最后，采用基于分析的策略和动态的策略的组合经常是一个好的思路。对于分析式测试来说，测试分析、设计和实现在测试执行之前进行。对于动态测试来说，相当多的测试分析、设计和实现工作发生在测试执行阶段。测试经理必须明白哪些测试用例是由分析策略引起的，哪些是由动态策略引起的。

## 2.5.1 测试执行

测试团队收到测试对象，并且满足了入口准则，测试执行就可以开始。测试经理需要掌握好这两个事项，以防过早进入一个测试级别产生不必要的耽误。

在测试执行时，测试人员按照测试规程执行测试。按照策略，经常会允许测试人员手动进行一定量的探索性测试。这个思想其实也就是测试管理的一种方法，因为测试经理要保证这个测试刚好达到某个期望程度，不多也不少。测试经理必须确保对测试采取了合适的控制，保证从探索性测试过程中获得必要的信息。因为探索性测试没有在测试规程中定义，如果不抓住额外的信息，就不能继承性地追溯测试依据。

当测试人员执行测试规程时，他们会将实际结果和预期结果相比较。测试经理要保证将测试结果错判率限制在可以容忍的小范围内。如果某个活动的结果没有通过但却标为通过那就是失效的正面结果。如果某个行为的结果通过了却标为没有通过就是所谓的失效的负面结果。以上两种情况都代表产生正确信息的测试过程的失效。所以测试经理



应该记录下这种事件的数量然后努力将数量减小。

当实际结果和期望结果不匹配时，就说明测试人员已发现到异常。这是识别和报告事件的第一步。避免失实的正面结果，测试人员需要仔细地调查事件，然后用正确的怀疑精神对待这些测试本身。一个重要的测试管理问题是：是否失实的正面结果的数量与过度的未受控的测试依据改变有关，比如说改变需求规格说明。

当测试人员执行测试时，他们应该记录测试结果。测试经理应当保证这些日志能够充分记录相应测试和测试事件细节。如果在测试中发生导致测试执行滞缓或者停止的事件时日志更加重要。测试经理应该准备随时被管理层要求对测试执行的延期做出解释，因为测试执行处在项目的成功发布的关键。测试经理需要保证有足够的配置管理，这样测试团队才可以清晰地将日志和测试中的特别测试目标的版本记录并联系起来。当然，测试经理应当保证测试日志包含必要的数据并生成测试结果的度量报告。

测试日志的详细程度和测试文档的详细程度一样，依据不同测试和策略级别而各有不同。在早期的测试级别，比如组件测试，日志会很少。在后期的测试级别，特别是规则标准制定后，日志会很多。

## 2.5.2 测试执行前置条件的案例学习

我们来看一下测试执行前置条件的例子。从图 2-3 中您可以看到 RBCS 所执行的一个测试项目的入口准则，这些入口准则集中了以下 3 个典型的测试前置条件：

### ISTQB 词汇表

测试日志：按时间顺序记录测试执行相关细节的文档。

测试记录：将测试进行执行并获得测试日志的记录过程。

- 测试环境的准备就绪。
- 被测试系统的配置和发布管理。
- 缺陷管理和测试管理系统的准备就绪。

1. 存在合适的缺陷跟踪和测试跟踪系统。
2. 所有组件由正式的自动化配置管理和发布管理所控制。
3. 运行团队已经配置好了系统测试服务器环境，包括所有目标硬件组件和子系统，而且测试团队得到进入这些系统相应的权限。

图 2-3 测试执行入口准则案例研究

另一个典型的测试前置条件是测试用例、测试数据等的准备就绪，它们没有在测试计划中显性地进行度量。在这个项目中，我们遵循一个基于风险的分析式测试策略，所以我们在项目的早期，在进入系统测试之前，实现了测试用例。这样做其实也是各有利弊。

在系统测试开始之前，用户界面小组就已经在周末加班了。在那个疯狂的周末，他们改变了整个用户界面。但是他们没有将这次改变告知测试小组，既没有口头通知也没

有电子邮件通知。

所以，接下去的那个星期一，当我们来到办公室准备继续我们的测试时，我们发现测试执行过程中出现了很多和系统早期版本不一致的地方。从用户界面角度来看，我们也不知道所有测试都失败的原因。我们怀疑是配置管理方面的问题，于是所有测试小组，一共 8 个人，花费一整天研究这件事情，直到和用户界面小组在回廊的一次谈话才知道是系统已经发生了变更。

这个不受控制的变更对我们测试工作量的影响是双重的。最直接的是，我们损失了一整天以及 64 个人每人一小时用来跟踪这些我们起初认为是缺陷的假负面问题。长远和巨大的影响是我们大约使用了两个人周时的工作量来更新这些测试。

如果我再次管理这个测试项目，我知道会发生什么事情，那么我还会继续选择在项目的早期来实现测试吗？是的，我会的。原因很简单：早期进行的测试实现揭示了许多重要的需求规格说明中的缺陷。如果没有我们之前做的详细测试实现工作，就不可能发现这么多问题。

在同样的项目中，我们有一个 Beta 版的测试级别。正如在基础级大纲中讲述的一样，Beta 测试是对大众传媒产品的实际用户进行的验收测试。产品市场团队制定该 Beta 测试的工作量，在图 2-4 中您会看到市场职员定义的该项目的目标。

目标：

号召 500 名员工或其他人员组成的大众人员对我们产品和网络的稳定性进行测试，然后提供稳定并详细地对产品和服务的反馈。这个反馈信息可以帮助我们在将产品推广到全国之前对产品进行改进。Beta 测试使我们的产品能在市场正式的发行，因此它非常重要，所以每个人都要参与进来。从该项目获得的结果和反馈是确定我们在市场成功的基础。

图 2-4 Beta 测试的目标

由于该产品是为无电脑用户设计的，所以找到会使用典型 Beta 项目方法的人也是具有挑战性的。所以为了复制具有相似技能和电脑经验级别的目标市场，市场团队经常号召各职员让他们的朋友或者家人来做 Beta 测试人员。

如果您仔细地读一下摘要，会觉得这项工作的目标有两面性。首先，典型的 Beta 测试会找到测试团队在正式系统测试中很难发现的缺陷。不过从测试团队的角度来说，因为他们也不可能考虑到所有的情况，所以这种测试的最终目标对于整个测试团队来说也是很有价值的。

当然，Beta 项目也应该包含市场相关组件。市场团队可以从项目中收集相关证明和其他有用的市场担保信息。正因如此，应该是市场团队而不是测试团队来管理该项工作。

从图 2-5 中可以看到 Beta 测试人员制作的该项目测试日志实例。正如您看到的，这个看起来不像一个专业的测试日志，比如说由一组专业测试人员做的系统测试。要求志愿者加强测试标准是非常困难的，这些问题也会发生在用户验收测试日志和组件测试日志中。在这种情况下，和其他的很多 Beta 测试不同，他们可以从测试人员那得到大量的数据，但是，测试日志有时候也会因为模糊不清的注释而难以阅读。



包括新闻财经和体育内容，非常好。

我还没有找到缺陷，但是我刚刚开始做。有更新我将随时告诉您。

我只浏览了一下气象图，印象非常深刻。

你们气象内容需要做些改进。

- 1) 看起来它经常落后 5 个小时，所以它需要更加“紧跟时间”。
- 2) 能不能提供一个更精确的地区卫星视图，比如说我们生活的地方？

图 2-5 Beta 测试人员的结果日志

市场和技术支持团队会对测试结果进行比较，尝试理解这些结果并和我们在正式测试时发现的问题进行对比。这项工作不是最终的或者具有决定性的，但是这种做法确实非常好。特别是进行用户验收测试的时候非常有用，因为需要将结果和系统测试结果进行比较。

### 2.5.3 测试标准 BS 7925/2

英国标准委员会制定了 BS 7925/2 标准。它有两个主要部分：测试设计技术和测试度量技术。它包括大范围的测试设计技术，包括黑盒、白盒测试等。基础级大纲涉及的黑盒技术在这里也有相应内容，那就是等价类划分、边界值分析以及状态转换测试。还有一种叫做因果图的黑盒测试技术，也就是决策表的图像化版本，最后还有语法测试。

对白盒测试技术而言，它包括语句测试和分支/判定测试。这些在基础级大纲中也有涉及。它也包含其他一些白盒测试技术，这些技术在基础级大纲中只是简单地涉及或者根本没有讲到，它们是数据流测试、分支条件测试、分支条件组合测试、改进的条件判定测试以及 LCSAJ（现行代码序列和跳转）测试。

还有两个部分的内容，它们是“随机测试”和“其他测试技术”，随机测试在基础级大纲中没有涉及，也不会在这本书中涉及，尽管我们会在技术测试分析这章讲述。关于其他测试技术的部分没有提供任何例子，而只是谈论如何选择这些测试的规则。

如果打算参加高级测试经理模块的认证考试，您可能会说“等一下，这样太快了。对于认证考试来说我需要知道些什么？”基础级大纲中的测试技术，您最好都了解，这对高级测试经理认证考试来说，要求不算过分。

BS 7925/2 对每个测试度量元技术提供了一到多种覆盖率度量。编写该文档的机构确实让人感到很奇怪，因为他们没有准确地说明为什么覆盖率度量元没有像设计技术中一样被覆盖。但是，从 ISTQB 的基础级测试过程来看，也许这样更加容易。例如，我们的入口准则可能会需要一些特别的测试级别覆盖，比如说我们在做某些安全关键的航空软件时要遵守美国联邦航空管理标准 DO-178B（本书后面会讲述该标准）。所以，在测试设计阶段，我们会用到测试设计技术。在测试实现阶段，我们会用测试度量技术来保证充分的覆盖率。



除了以上两个主要部分,该文档也包括两个附录。附录 B 通过将前两个主要部分中的事项放在真实情景下再现而使之更加生动。附录 A 包括过程注意事项,这对我们的帮助似乎是最大的。它按照文档所给的测试过程讨论了测试项目的应用标准。

我们可以将该过程和 ISTQB 基本测试过程相对应。ISTQB 中的测试分析和设计等价于 BS 7925/2 过程中的测试规格说明。BS 7925/2 中的测试执行和 ISTQB 中的测试执行等价。注意,在 ISTQB 过程中测试执行作为测试实现和执行中的一大行为。ISTQB 过程也包括作为测试执行中一部分的测试日志,而 BS 7925/2 有一个独立的测试记录过程。最后,BS 7925/2 在最终步骤过程中会检查测试完成度。这个步骤大约相当于 ISTQB 中的评估测试出口准则和报告。

结束本章内容的时候,我们来看一下 ISTQB 基本测试过程中与测试实现和执行对应的度量元和度量。当然不同的人有不同的度量元标准。

在测试实现过程中的典型度量元是测试环境配置的百分比、测试数据记录载入百分比和自动测试用例百分比。在测试执行中,典型的度量元会包括覆盖的测试条件、执行的测试用例等。

一个工作分解结构对于衡量我们是否按照估算和进度进行工作也是非常有用的。

注意我们这里讨论的是衡量这些过程完成度的度量元,例如,我们自己完成的进度。您应该使用一组不同的度量元来制作测试报告。

## 2.6 评估出口准则和报告

### 学习目标:

(K2) 汇总在测试过程中收集的必要信息以支持精准的报告和评估出口准则。

出口准则的评估和报告测试结果属于测试管理活动。我们会介绍一些方法用来分析、展示和报告这些测试结果,并将这些方法作为测试过程监控活动的一部分,这些会在第 3 章详细介绍。

从测试过程来看,我们需要考虑两个主要方面。首先,我们需要收集必要信息来支持测试管理层的报告结果。其次,我们需要根据测试完成情况测量进度,另外需要发现计划之间的偏差。

为了评估测试是否达到出口准则,以及生成报告所必需的信息。我们可以测量以下测试执行过程的属性:

#### ISTQB 词汇表

**出口准则:** 由各利益相关者认可的、正式结束一个过程所需的一系列具体条件。出口准则是预防未完成项目被人误认为已经完成。出口准则用来报告差距和计划何时结束测试。

- 测试条件、用例或者计划、执行、通过和失败的测试规程的数量。

- 按照严重程度、优先级、状态或者其他因素分类的全部缺陷。
- 提议、接受和测试的变更请求。
- 计划和实际的成本、进度和工作量。
- 减轻和剩余的质量风险。
- 由于事件的阻碍导致的测试时间损失。
- 确认和回归测试的结果。

一个工作分解图结构在判定是否正按照估算和进度表进行时非常有用。

如图 2-6 所示是 4 个典型的测试结果报告图形。我们简要描述一下这 4 幅图并讲述测试过程中创建的数据。

在左上角的是开/关，或者缺陷集合图表。这个图表记录的缺陷发现率和缺陷记录日志直接决定了该项目过程中从发现缺陷到缺陷解决的时间。这个图表记录了每个缺陷的开启日期和结束日期。

按照顺时针方向走，右上角是测试过程表。这个表用来测量每天的测试过程是否稳定和有效。该表需要记录每天计划和实际的测试小时数。

右下角是测试完成图。这个图用来记录完成计划测试用例的数量以及成功和失败的用例数量。这个图需要显示我们每天计划运行的所有测试用例的数量，实际运行的测试用例的数量，成功的实际测试用例的数量和失败的测试用例的实际数量。

最后，在左下角的是测试覆盖率图。这个图用来衡量我们是否覆盖了测试依据。在这个例子中，也就是每个主要风险域所发现的相关缺陷和质量风险。这个表按照测试依据的要求使测试用例和缺陷报告可以被追溯和分析。

以上 4 个图以及更多类似的图我们会在第 3 章详细讲述。

本小节结束前，我们来看一下 IEEE 829 对于测试文档的标准定义，在这个案例中，也就是测试总结报告的模板。一个测试总结报告描述了一个测试的阶段或者级别的结果。IEEE 829 模板包括以下部分：

- 测试总结报告标识符。
- 总结（比如，测试了什么、结论是什么等）。
- 差异（和计划、用例、过程相比较）。
- 综合评定。
- 结果总结（最终度量元）。
- 评估（对每个测试项使用通过/失败测试标准）。
- 活动总结（资源使用、效率等）。
- 批准。

测试经理在测试执行的时候可以发布这些总结，以此作为项目状态报告或者会议的一部分。这些总结也可以用在测试级别的后期作为测试结束活动的一部分。

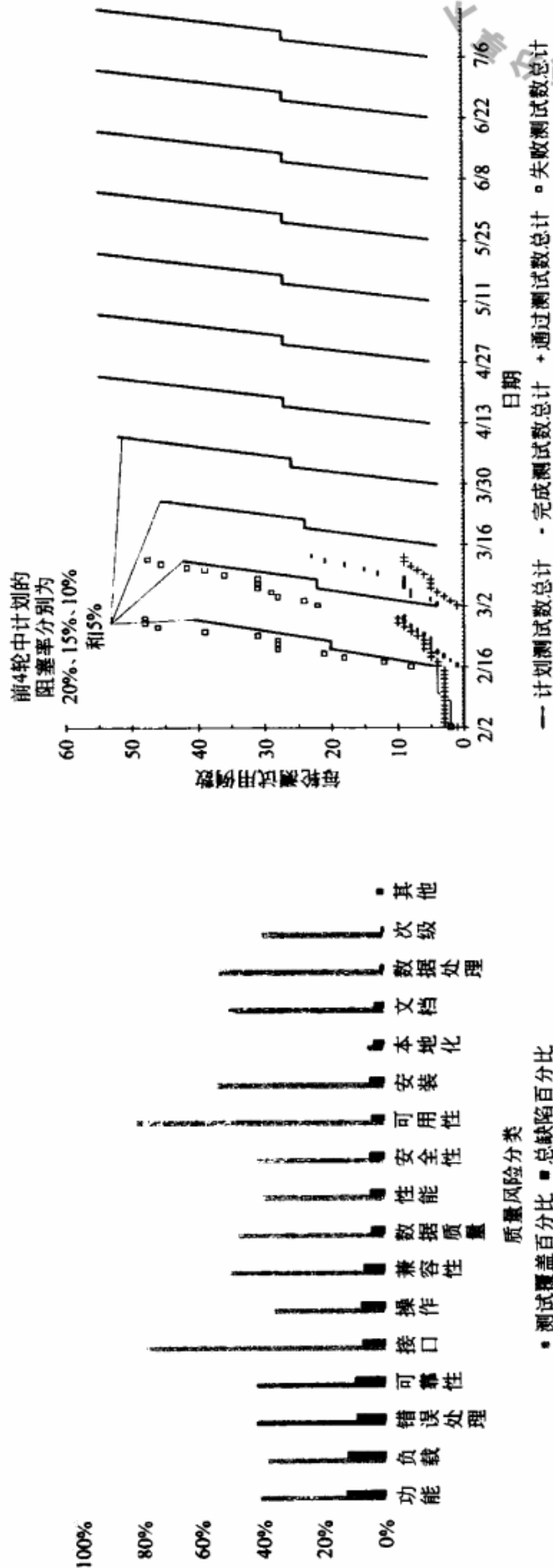
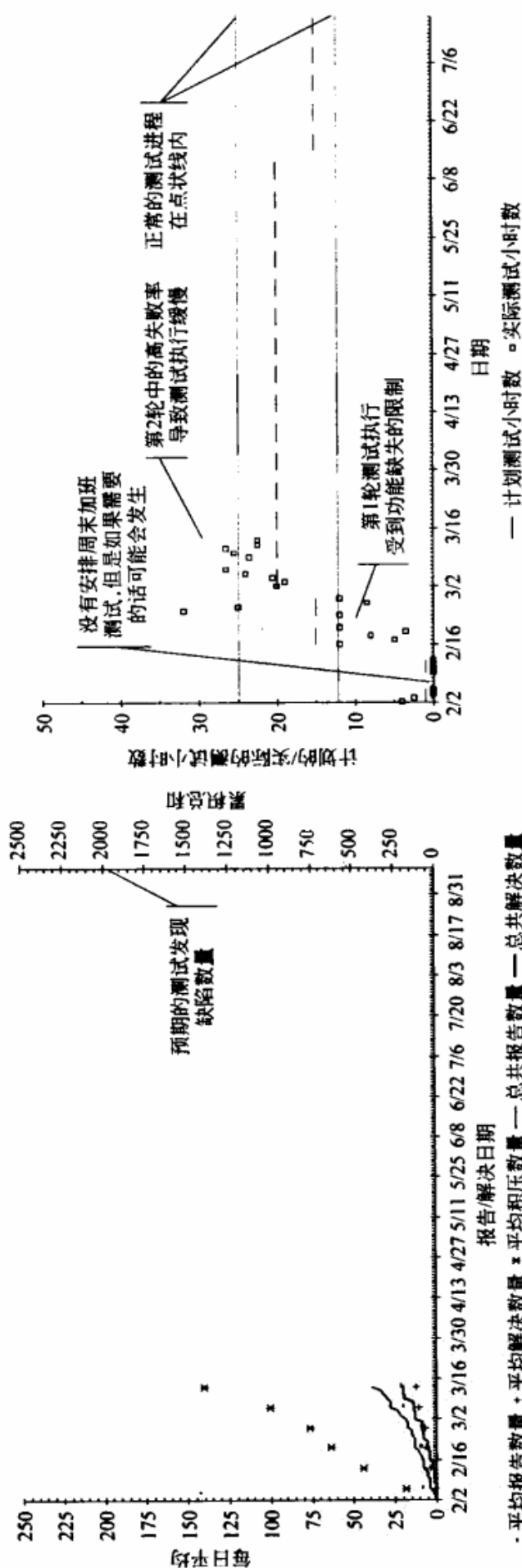


图 2-6 测试结果报告仪表盘



### ISTQB 词汇表

**测试结束：**从已完成的测试活动中收集数据，汇总经验、测试件、事实和数量的测试结束阶段，包括对测试件的最终处理和归档，以及测试过程评估（包含测试评估报告的准备）。

**测试概要报告：**该文档概括了测试活动和结果，也包含了和出口原则相对应的测试项的评估。

## 2.7 测试结束活动

### 学习目标：

(K2) 概要介绍 4 种主要的测试结束活动。

(K3) 归纳在测试结束阶段学到的经验教训，用来发现可改善或重复的部分。

测试结束工作在测试过程的末期经常会被遗忘，但其实它是非常重要的，它可以获得对测试团队、大项目团队和整个组织非常有用的剩余价值。测试结束工作有 4 种主要形式：

- 保证测试圆满结束。
- 交付测试工作产品。
- 参加项目回顾会议。
- 归档测试工作产品。

就像上面提到的，测试结束工作经常被遗忘，或者至少一个或者多个这样的活动经常被遗忘。在某些情况下，结束工作会被简单省略，有时是由于测试或者项目团队过早地退出，或者由于后续项目的进度压力重新分配资源导致无法开展这些工作，或者只是因为团队的解散。对于基于合同的开发而言，合同中应该指明结束活动以保证它们不会被忘记。

让我们依次来看一下上述几个活动。首先是简单的检查以保证我们确实做完了测试。考虑到管理测试的复杂性，特别是对于大的分布式项目，导致很容易忘记某些事情。聪明的测试经理在测试接近结束时，会反复检查测试计划来保证所有计划的事项都已经做完。他们会检查测试跟踪信息来保证所有计划的事项要么是运行结束得到结果，要么是被谨慎地忽略（该忽略得到大家的同意）。他们检查所有已知的缺陷确保其被修复（包括修复以后的确认测试）、延期（从项目团队得到的合适的准许）或者被接受作为长期的缺陷限制。

接下来是寻找机会重用测试工作产品。在产品发布之后，这些工作产品对项目里的其他人、支持团队，甚至客户都非常有用。例如，我们告诉用户、客户和支持人员关于已知延迟或者接受的缺陷。我们会把测试和测试环境转交给进行维护测试的人员。我们可将自动或者手动回归测试包交付给客服，如果他们将我们的系统集成进入更大的综合

系统的话。

最通常的测试结束活动的方式是使用项目回顾来记录在项目中学到的经验和教训,以更好地计划管理未来项目中可能的事件和问题。例如,在项目回顾的分析中,我们发现很多未预料到的缺陷。这个发现可以帮助我们在未来的项目中质量风险分析的参与人数增加。

我们可能会发现在某些领域的估算完全失真。这个就需要我们去评估导致偏离的根本原因。偏离的原因可能包括比期望更多的缺陷、比期望更多的测试工作等。我们在之后会采取一些措施来解决这些问题。

我们可能会发现缺陷趋势和起因会引导我们更早发现缺陷、减少缺陷数量,或者杜绝缺陷产生的机会。我们可以定位到测试或者整个项目中效率低、效果差的地方,然后可以确定过程、团队或者工具方面需要改进的地方。

记住在测试中需要经常运用回顾,贯穿整个项目或者更大的组织中。问题会批量出现,而绝不是孤立的。当您有机会进行改进时,会发现在目标改进团队之外经常会有促进者和阻碍者。

最后,测试结束活动经常还包括信息产品的安保工作。测试结束后会发布中间或者最终的测试结果、测试日志文件、测试状态报告和其他文档和工作产品。我们应当将这些放到配置管理系统,而配置管理系统必须将不同的测试工作产品与被测系统联系起来(按照名字和版本)。我们应该取得测试计划和项目计划,并将其存档,同时保存测试规程、测试数据和测试结果。

### 2.7.1 测试结束的两个案例

我们以该网络相关项目为例,讲述项目后期将测试工作产品交付给另一方并体现其价值。在这个项目结束的时候,也就是在产品发布之前,我让一个测试人员将所有已知的项目缺陷和技术支持经理以及技术支持团队的主要参与者进行沟通。这个报告包括失败测试用例的详细信息,以及测试中失败的特定步骤,和期望结果不一致的实际结果,如果可能的话,提供我们识别的任何补救措施。它也要包括对延缓或者未解决缺陷的完整性回顾。根据技术支持经理的经验,这样做会节省多达 300%~400%的时间,以解决询问已知失败测试和缺陷的技术热线和电话所用的时间。

一个偶然的机会,我的测试团队和我一起尝试清理一个失败项目的第二轮的测试。第一步是分析测试结果以进行项目回顾。图 2-7 是我创建的一幅图,作为回顾分析的一部分。这个测试非常差劲,通过分析之后揭示了其中的很多原因。

图中我们可以看到不同的缺陷报告被打开的次数。如果一个缺陷被发现、报告,然后被开发人员修复,关闭再也不出现,那么这个缺陷的计数度量元为 1。如果一个缺陷被发现、报告、修复好然后在确认测试中重新打开,然后才被修复,关闭不再出现,那么这个缺陷计数度量元就为 2。相应的,如果这个缺陷被发现、报告、修复,在确认测试中被关闭,但是由于回归测试,又被打开,而且只有第二次是最终解决的,那么这个缺陷计数度量元也是 2。顺便提一下,这个范围是对数级别的,这样就可以保证大家的注意力集中于这个图的关键点,即大量的缺陷数量。



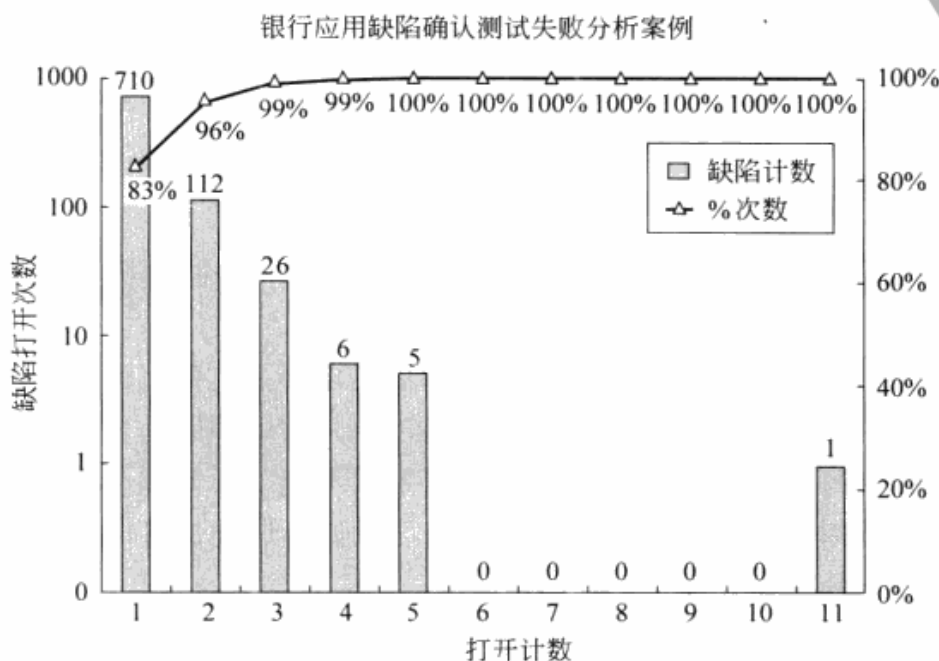


图 2-7 确认测试的高失效率图

本图的关键点是大量的缺陷报告需要反复处理。正如您所见，整整 1/6（大约 17%）的缺陷报告至少被重新开启一次。这是个大问题吗？假定每个缺陷报告需要至少 1 个小时来处理，那么我们可以看到，从柱状图第二行开始自左至右表示反复处理的缺陷总共浪费的时间：

- 有 112 个缺陷报告被开启了两次。第二次的开启完全是浪费时间，所以我们在两次开启问题的缺陷报告上浪费了 112 个小时。
- 有 26 个缺陷报告开启了 3 次。第二次和第三次都是浪费时间。所以您可以说在 3 次开启问题的缺陷报告上浪费了 52 个小时。
- 有 6 个缺陷报告开启了 4 次。第二次、第三次和第四次都是浪费时间。所以您可以说在 4 次开启问题的缺陷报告上浪费了 18 个小时。
- 有 5 个缺陷报告开启了 5 次。第二次、第三次、第四次和第五次都是浪费时间。所以您可以说在 5 次开启问题的缺陷报告上浪费了 20 个小时。
- 有 1 个缺陷报告开启了 11 次。第二次到第 11 次都是浪费时间。所以您可以说在 3 次开启问题的缺陷报告上浪费了 10 个小时。

由于反复处理这些缺陷报告，总计超过 200 个人小时或者 5 个人周的时间被浪费。所以这个故事告诉我们一个道理，那就是一次性解决，而且要保证正确。

在结束本章节的时候，我们来看一下测试结束活动的度量元和度量。为了测量测试过程的完成度，我们通常测量以下几点：

- 测试执行中运行测试用例百分比。
- 登记入册的测试用例百分比。
- 自动化的测试用例百分比。
- 回归测试用例百分比。
- 延缓缺陷报告百分比。



- 归档工作产品百分比。

一个工作的分解结构对于我们将实际值与估算和计划进度进行比较非常有用，对于项目回顾来说也是有益的。

## 2.7.2 测试结束活动练习

在 HELLOCARMS 系统的第一次发布之后，举办了一个回顾会议。通过研究结果，发现了以下问题：

- 反向住房按揭贷款功能中报告的 27% 的缺陷以不是实际问题为由被拒绝接受。
- 额外人寿保险功能中报告的 25% 的缺陷以不是实际问题为由被拒绝接受。
- 对贷款文件打印系统中报告的 37% 的缺陷以不是实际问题为由被拒绝接受。

该团队平均的缺陷报告拒绝率是 5%。我们可以认为这个是可以接受的，但是任何超过 10% 的情况都是不可接受的。

为第二次及其以后版本的发行列出至少 5 种可能的改进方法来进行调查。

## 2.7.3 测试结束活动练习参考答案

提供的列表应该包括下面的某些条目：

- 通过培训，招募或者用户帮助来增加团队中的反向住房按揭贷款专家。
- 评审特定的与被拒绝的反向住房按揭贷款缺陷相关的需求是否模糊不清或者是否错误。
- 对现场报告中被拒绝的反向住房按揭贷款缺陷的高发生率进行检查。
- 通过培训、招募或者用户帮助来增加团队中的人寿保险专家。
- 评审特定的与被拒绝的人寿保险缺陷相关的需求是否模糊不清或者是否错误。
- 对现场报告中被拒绝的人寿保险缺陷的高发生率进行检查。
- 通过培训、招募或者用户帮助来增加团队中的贷款文件打印系统专家。
- 评审特定的与被拒绝的贷款文件打印系统缺陷相关的需求是否模糊不清或者是否错误。
- 对现场报告中被拒绝的贷款文件打印系统缺陷的高发生率进行检查。
- 建立一个开发者会议来讨论为什么这些问题被拒绝了，然后集思广益改进两个团队之间沟通的方法。
- 进行缺陷回顾，这样确保每个缺陷报告在进入追踪系统之前被一个或者多个测试人员检查。

您可以想出其他合适的办法。但请记住，进行改进时的重要事项就是使用数据。

## 2.8 认证考试模拟题

在结束本章之前，来看一下试题的例子以加深对所学知识的理解并准备 ISTQB 高级测试经理认证考试。

1. 假设您是某项目的测试经理，该项目创建一个可编程恒温器来控制家庭用途的中央供

热系统、空气流通设备和空调系统。这个项目采用顺序生命周期模型，具体地说就是V模型。您的测试团队遵守基于风险的测试策略并在测试执行中结合反应式测试策略，为捕获风险分析时遗漏的风险项提供保障。

找出以下关于测试计划正确的论述：

- A 测试计划应当在项目早期就包含风险分析
  - B 测试计划应当把需求规格说明列为风险分析的输入条件
  - C 测试计划应当将特定缺陷指派给特定的开发人员
  - D 测试计划应该讨论如何将反应式测试和测试执行进行集成
  - E 测试计划应当定义决定项目团队年终红利的所有度量元
2. 从下面哪一份测试文档中可以找到某个测试级别开始测试执行的前提条件？
- A 测试计划
  - B 测试设计规范说明
  - C 事件报告
  - D 项目计划
3. 在测试执行的时候，邀请用户介入的最可能的原因是什么？
- A 他们的应用领域业务知识
  - B 他们的技术专长
  - C 他们的测试专长
  - D 他们的管理专长
4. 您是某银行质量评估小组的经理，负责银行项目的独立测试。您刚完成一个项目，该项目将3个商业成熟系统集成成为银行的应收账款处理系统。项目过程中，您发现其中一个分包商的子系统存在重大缺陷，尽管该系统与其他两个系统相比，功能数目相同，复杂度类似。基于以上情况，请找出合理的选项用来改善后续项目中同样有该分包商参与的项目的测试过程：
- A 按照该分包商在该项目中提交的缺陷数量事后对其要求罚款
  - B 对所有提交系统进行验收测试，对该分包商的测试要特别的严格
  - C 取消和该分包商的合同将其放入业务黑名单
  - D 建议该分包商的开发人员参加培训以提高他们代码的质量

## 第3章

# 测试管理

“计划是我们给你的团队一个测试发布，你们对它运行一轮测试。如果你们发现任何缺陷，我们将会周末修复然后给你们一个最终发布版本。在你们确认一切都正确之后，我们就准备发布。”

——一个过分乐观的开发经理向作者如此描述了他对测试执行的预期。实际上测试执行结果和预期有些不同，他们发现了超出预期许多的缺陷。

ISTQB 高级大纲第3章是关于测试管理。本章讨论了整个测试过程中的测试管理活动，并且介绍了测试中的风险管理。本章包括以下11个部分：

1. 概述。
2. 测试管理文档。
3. 测试计划文档模板。
4. 测试估算。
5. 测试计划进度。
6. 测试进度监视与控制。
7. 测试的商业价值。
8. 分布式测试、外包测试与内包测试。
9. 基于风险的测试。
10. 失效模式与影响分析。
11. 测试管理相关问题。

让我们看一下每个部分是如何与测试管理紧密结合在一起的。

### ISTQB 术语

**风险分析：**评估识别出的风险以估计其影响和发生的可能性的过程。



## 3.1 概 述

### 学习目标:

能回想起该部分内容即可。

本章主要讲述关于测试管理的内容。因此，它对于参加高级测试管理认证考试的报考者相当重要。相比较大纲中的其他章节，你应该花更多的时间在这一章上。

本章也涵盖了测试经理所用的测试文档模板。这部分主要关注 IEEE 829 标准。

我们将不会完全按照大纲中的顺序来讲述本章节的内容。我趋向于遵从基于风险的分析测试策略。在这样的策略中，风险分析会优于除了设定测试目标和测试上下文关系以外所有其他的测试任务。

所以我们将要从大纲的 3.9 节和 3.10 节开始，这两部分都是关于风险分析的。因为质量风险分析练习要采用大纲 3.10 节中描述的“失效模式和影响分析”技术方法，所以我将结合这个技术方法和大纲中 3.9 节的内容。当结束这两部分内容后，我们会依次讲述第 3 章中剩下的部分。

基于同样的理由，我将大纲 3.2 节和 3.3 节的内容进行了合并。因为大纲 3.2 节中的关于测试计划部分对应的练习需要使用到在 3.3 节中讨论的 IEEE 829 测试计划模板。

为了更好地保持与大纲的一致性，我保留了大纲的章节名。但是本书中章节的顺序可能和大纲中的有所不同，有些章节可能会涉及大纲中多个章节的内容。

## 3.2 基于风险的测试与失效模式和影响分析

### 学习目标:

(K2) 解释基于风险的测试如何在多个方面考虑风险因素。

(K4) 区别产品风险和项目风险，并能基于这些风险制定合理的测试策略和测试计划。

(K3) 从测试人员的角度，根据 FMEA (failure mode and effect analysis) 方法对一个产品进行风险分析。

(K4) 汇总项目关键成员对项目风险的不同观点，使用汇总后的结果开展降低风险的测试活动。

(K2) 描述需要不断迭代的风险管理的特性。

(K3) 将基于风险的测试策略转化为实际的测试活动，并在测试过程中监控其效果。

(K4) 分析和报告测试结果，包括确定和报告遗留的风险以帮助项目管理人员做出正确的决断。

(K2) 描述 FMEA 的基本概念，通过例子解释其在项目中的应用和对项目的益处。

风险是一种负面的或者不想要的结果，或者事件发生的可能性。一个具体的风险是

指会导致客户、使用者、参与者或其他利益相关者降低对产品质量或项目成功看法的任何可能发生的问题的可能性。

在测试中，我们主要关心两类风险。第一类风险是产品风险或称为质量风险。当潜在问题主要影响产品本身质量的时候，该潜在问题被称为产品风险。对于产品风险我自己经常使用的一个同义词是质量风险。一个质量风险的例子，比如一个可能导致系统在正常操作时崩溃的可靠性缺陷。

#### ISTQB 术语

**产品风险：**与测试对象有直接关系的风险。

**项目风险：**与测试项目的管理与控制相关的风险。例如：缺乏配备人员、严格的限期、需求的变更等。

**风险：**将会导致负面结果的因素。通常表达成可能的负面影响。

第二类风险是项目风险或称为计划风险。当潜在问题主要影响项目的总体成功时，该潜在问题被称为项目风险。有些人也把项目风险称为计划风险。一个项目风险的例子，比如可能因为人手短缺导致项目延期。

并非所有的风险都一样重要。有许多方法可以区分风险的级别。最简单的方法是考虑以下两个因素：

- 问题发生的可能性。
- 问题发生后的影响。

问题发生的可能性主要是从技术的角度考虑，例如所使用的编程语言、连接的带宽等。问题发生后的影响主要是从商业角度考虑，例如遭受的财务损失、受到影响的用户或客户数量等。

在基于风险的测试中，我们使用风险分析中已识别的风险项和每个风险项的风险级别来指导我们的测试。实际上，在真正的分析式的基于风险的测试策略中，风险是测试的首要基础。

风险能通过各种不同的方式指导测试，其中有3种方式比较常用：

- 首先，在所有测试活动中，测试分析师和测试经理根据其质量风险项的风险级别按比例分配人力。测试分析师选择和风险级别相匹配的测试技术。测试经理和测试分析师按照风险级别执行测试活动，也就是说首先处理最重要的质量风险，直到最后才处理那些次要的风险。最后，测试经理、测试分析师应和项目组所有成员一起保证缺陷的优先级设定和解决方案适应其风险级别。

#### ISTQB 术语

**风险级别：**风险的重要性，由风险的影响和可能性定义。风险级别能用于决定测试的强度。风险级别既能用定性的词（比如：高、中、低）表示，又能用定量的词表示。

**风险管理：**对风险进行标识、分析、优先计划和控制所应用的系统化过程和实践。



- 其次，在测试计划和测试控制中，测试经理对所有重要的已识别的项目风险实行风险控制。项目风险级别越高，实行的控制越严格。马上我们将会讨论风险控制的各个选项。
- 第三，测试经理和测试分析师根据剩余的风险报告测试结果和项目状态。例如，哪些测试还没有被运行或者被跳过了？哪些测试已经被运行了？哪些测试已经通过了？哪些测试失败了？哪些缺陷还未被修复或者未进行再测试？测试和缺陷如何与风险关联？

当采取真正的分析式的基于风险的测试策略时，风险管理不应当仅仅出现在项目启动时期。刚刚提到的3种管理风险的方法以及其他可能需要的方法应当贯穿整个项目生命周期。说明确些，我们应当尝试通过运行测试和寻找缺陷来降低质量风险，并通过风险缓解，必要的时候通过应急方案来降低项目风险。项目中的风险和风险级别应当定期地根据新的信息进行重新评估。这可能会导致重新划分测试和缺陷优先级，重新分配测试人力和其他的测试控制活动。这些会在本章节以后的部分进一步讨论。

为了帮助大家更好地理解基于风险的测试，测试有时被比喻成保险。在日常生活中，一个人购买保险是因为担心某些潜在的风险。你不会为那些你并不担心的风险购买保险。因此，我们应当着重测试那些令人不安的地方和缺陷而忽略那些你并不担心的地方。

以上比喻有一个潜在的误导，保险专业人士和精算师能够使用有效的统计数据来进行定量的风险分析。而通常来说，因为缺乏像保险公司所拥有的那些数据，基于风险的测试则依赖于定性分析。

在基于风险的测试中，你必须意识到风险来源有许多种。有存在于一些系统的安全性风险；有存在于大部分系统的商业和经济上的风险；有存在于许多系统的私密性和数据安全风险；也有技术、组织和政治上的风险。

### 3.2.1 基于风险测试的特性和好处

分析式的基于风险的测试策略包括什么呢？它有哪些特性和好处呢？

首先，分析式的基于风险的测试策略将风险级别和测试人力进行匹配。风险级别越高，投入的测试人力越多。这不仅仅包括测试执行所耗人力，也包括测试设计和实现所花费的人力。我们会在本章的后半部分讨论相关方法。

其次，分析式的基于风险的测试策略将风险级别和测试的顺序进行匹配。对风险较高的部分进行测试会发现更多的缺陷或者测试系统中相对更重要的地方，或者两者兼有。因此风险越高，就应该越早安排测试。首先寻找那些最可怕的东西，这是我经常告诉测试人员的一个经验法则。我们会在本章的后半部分讨论如何做到这一切。

因为以上人力分配和测试顺序的安排，随着测试的进行，总的剩余质量风险级别会可预见地降低。通过维护测试与风险之间、已定位的缺陷与风险之间的可追溯性，我们能够根据剩余的风险报告测试结果。当继续测试的风险超过了宣布测试结束的风险时，项目利益相关者可以决定宣布测试结束。

既然我们可预见到剩余风险将会降低，这意味着我们可以根据风险高低选择测试。当进度被压缩时，我们应当缩小测试范围。对于项目利益相关者来说，按照风险高低来



选择测试范围是一种可接受的方法。

### ISTQB 术语

**风险识别：**使用技术手段（例如，头脑风暴、检验表和失败历史记录）标识风险的过程。

基于以上理由，分析式的基于风险的测试策略比分析式的基于需求的测试策略更健壮。纯粹分析式的基于需求的测试策略对于每个需求至少需要一项测试，但是该策略里并没有提到需要多少测试才能智能地应对项目的限制，也没有提到运行测试的具体顺序。在纯粹的分析式的基于需求的测试策略里，整个测试执行过程中的风险降低是无法预测和不可度量的。因此，当项目利益相关者询问我们是否能够安全地削减或压缩测试时，我们无法简单地用分析式的基于需求的测试策略来描述剩余的风险级别。

这并不意味着当我们使用分析式的基于风险的测试策略时可以忽略需求规格说明。相反，在允许的情况下我们仍然使用需求规格说明、设计规格说明、市场要求、技术支持或者服务台（help desk）数据以及无数其他的输入来描述风险识别和分析过程。然而，如果我们没有这些信息或者我们发现这些信息的有效性不足，我们依旧可以使用利益相关者提供的风险识别和评估过程来计划、设计、实现和执行测试。因为我们减少了对像需求收集和 design 这样的（我们可能无法控制的）上游过程的依赖，所以这种能在缺少或没有文档的情况下仍然生存的能力使得分析式的基于风险的测试策略比分析式的基于需求的测试策略来得更为健壮。

综上所述，分析式的基于风险的测试策略并非十全十美。和其他所有分析式的测试策略一样，在项目开始阶段我们不会获得完美风险评估所需的所有信息。即使通过阶段性风险重估（将在本章后半部分讲述），我们也会遗漏一些重要的风险。因此，和其他所有分析式的测试策略一样，分析式的基于风险的测试策略应当在测试实现和执行阶段混合使用反应式策略，从而发现在风险评估阶段遗漏的风险。

让我就经常遇到的测试问题具体地谈一谈分析式的基于风险的测试是如何来解决的。

首先，作为测试人员我们经常会面临巨大的时间压力。我们很少有足够的时间来运行想要运行的测试，而当运用基于需求的测试策略时这个问题尤为明显。最终所有测试都是固定时间长度的。基于风险的测试提供了一种在软件生命周期的任意阶段都可以对测试进行优先级排序和筛选的方法。

当我提到所有测试都是固定时间长度的时候，我要提我们在挑选合适的测试覆盖率时面临的挑战。如果我们按可测试的百分比来度量测试覆盖率，那么任何数量的测试产生的覆盖率都是 0%，因为对于任何实际系统而言可运行的测试数量是无限的。所以，基于风险的测试提供了一种能够从无限大的我们能够运行的测试集合中挑选一个合适子集的方法。

另外，我们也经常遇到规格说明缺少或者被测试对象质量差的情况。通过让利益相关者参与决定哪些需要测试，哪些不需要测试以及测试多少，基于风险的测试能够帮助

我们识别和弥补可能给测试带来大麻烦的文档漏洞（例如需求规格说明）。同时，它也可以使得其他利益相关者意识到哪些需要测试（测试多少）、哪些不需要测试这个难题。

回到时间压力的这个问题，在测试执行阶段，这个问题不仅很严重，而且会不断升级。我们经常在测试执行的开始阶段，甚至中途被告知需要压缩测试时间。基于风险的测试提供了智能地筛选测试的方法，同时它也提供了一种和项目利益相关者讨论压缩测试所带来风险的渠道。

最后，在测试执行结束时，我们需要帮助项目利益相关者做出明智的发布决定。基于风险的测试不再是强迫所有人依靠例如缺陷数和测试数等不充足的策略性度量来做发布决定，而是和利益相关者一起决定剩余风险的可接受级别，从而做出合理的发布决定。

### 3.2.2 基于风险测试的历史

分析式的基于风险的测试策略是如何形成的呢？了解这段历史可以帮助我们理解这些策略的现在和未来。

在 20 世纪 80 年代早期，Barry Boehm 和 Boris Beizer 各自独立地研究了软件开发相关风险的课题。Boehm 提出了风险驱动的螺旋开发生命周期，这已经在基础课程大纲中讨论过了。这种方法按照风险的高低顺序开发架构和设计，从而减少了项目开发后期大幅变化和进入死胡同的风险。

Beizer 提出了风险驱动的集成和集成测试。换言之，仅仅按照风险的高低顺序进行开发是不够的，我们也需要按照风险的顺序进行集成和测试。<sup>1</sup>

如果仔细推敲 Boehm 和 Beizer 的观点含义，你就能够发现这些思想就是迭代和敏捷生命周期的初期形式。

在 20 世纪 80 年代中期，Beizer 和 Bill Hetzel 各自独立地宣称风险应该是测试的主要驱动力，这同时包括了测试人力和测试顺序。然而，虽然大致描述了这个思想，他们却并没有详细制定明确的方法论。我并不是要在这里批评他们。在那个时候，能让测试人员意识到风险似乎就已经足够保证基于风险的测试了。<sup>2</sup>

然而，情况并非如此。有些测试人员已经按照他们自己对风险的想法来决定测试范围和优先级。因为某些原因，这导致测试变成了信息闭塞、被动反应式的缺陷搜索活动，我们会在后面讨论这个问题。发现很多缺陷本身没有问题，但是追求发现尽可能多的缺陷并不是一个很恰当的测试目标。

因此我们需要更加结构化的方法来保证风险的系统化探索。这把我们带到了 20 世纪 90 年代。Rick Craig、Paul Gerrard、Felix Redmill 和我分别寻找能把基于风险的测试这个概念系统化的方法。我不知道 Craig、Gerrard 和 Redmill 怎么想，但是基于前面提到的理由，我知道我自己已经对基于需求的测试策略失望了。所以我们 4 个人，或许还有其他人，各自独立并行地开发出质量风险分析和基于风险测试的类似方法。本章节会学习到

---

1 参见 Beizer 的“Software System Testing and Quality Assurance”。

2 参见 Beizer 的“Software Testing Techniques”和 Hetzel 的“Complete Guide to Software Testing”。



这些方法。<sup>3</sup>

那么,我们现在在哪里?从2000年代的中期到晚期,测试人员采用不同的形式广泛地使用了分析式的基于风险的测试策略。虽然仍有一些人采用被误导的、被动反应式的、基于测试人员的缺陷搜索策略,但是许多测试人员正在使用分析式的方法来尽可能防止在测试的后期发现缺陷,这包括集中测试最可能出错和最重要的部分,根据剩余的风险报告测试状态以及当对于风险的理解发生变化后更好地进行反馈。在将这节的思想运用到实践中时,您可以加入我们的尝试。当您分析式的基于风险的测试以及哪里还需要改进有了更多的了解,建议您用文章、书、演示等方式将学到的知识与大家共享。

虽然我们还有许多需要学习的,但是这并不意味着分析式的基于风险的测试策略仅仅只是实验性的。它是已经被证明的具有实践性的策略。在软件项目无数的现状和限制下,我不知道还有哪种其他的测试策略可以做的和分析式的基于风险的测试策略一样好。特别是在结合了反应式的策略之后,这种策略是最佳选择。

另一种需要注意和研究的结合形式是分析式的基于风险的测试策略和所有现存生命周期模型的结合使用。我的同事已经在顺序的、迭代的和螺旋形的软件生命周期中分别使用过分析式的基于风险的测试策略。结果表明策略是否奏效和采用哪种软件生命周期无关。然而,策略必须适合于该软件生命周期。

除了在实践中继续学习以外,配套相应的支持分析式的基于风险的测试策略的测试管理工具是另外一个重要的步骤。现在有些测试管理工具已经结合了基于风险的测试实践。有些测试管理工具仍然不支持基于风险的测试。我鼓励那些工作在测试管理工具上的人们能够开发出支持这个策略的工具并且不断地改进它。

#### ISTQB 术语

**风险控制:** 为降低或控制风险到指定级别而做出决定和实施防范(度量)措施的过程。

**风险缓解:** 参见风险控制。

### 3.2.3 如何进行基于风险的测试

让我们回到如何执行基于风险的测试这个策略性问题上。首先总体地讨论一下风险管理,然后在剩下的部分里集中讨论基于风险的测试的具体内容。

风险管理包括3项主要活动:

- 风险识别,指出项目中的项目风险和产品风险。
- 风险分析,评估每个已识别风险项的风险级别,通常基于发生的可能性和发生后的影响。

---

<sup>3</sup> 关于我的方法的细节,参见我在“Critical Testing Processes”和“Pragmatic Software Testing”书中对非正式方法的讨论。关于Paul Gerrard的方法,参见“Risk-based e-Business Testing”。Van Veenendaal在“The Testing Practitioner”一书中讨论了他的非正式方法。



- 风险缓解（也被更恰当地称为“风险控制”，因为它包括了针对各类风险的风险缓解、风险应急、风险转移和风险接受策略）。

这些活动在某种意义上是连续的，至少在开始的时候是如此。风险识别首先开始，随后是风险分析。而一旦我们通过风险分析决定了风险级别，风险控制就开始了。然而，因为我们应当在项目中持续地管理风险，风险识别、风险分析和风险控制都是循环的活动。

每个人都有他自己对于项目风险管理的观点，包括哪些是风险、风险的级别和相应的风险控制。因此，应当让所有项目利益相关者参与风险管理。

因为专注于缺陷，测试分析师拥有风险管理特定的专业技术，他们应尽可能地参与风险管理。实际上在很多情况下，测试经理在主持质量风险分析的过程中都会得到测试分析师的关键支持。

让我们更详细地看一下这些活动。在正确的基于风险的测试中，我们需要识别产品风险和项目风险，我们可以通过以下方法识别这两种风险：

- 专家咨询。
- 独立评估。
- 使用风险模板。
- 项目回顾。
- 风险研讨会和头脑风暴。
- 检查清单。
- 以往的经验。

可以使用集成的单一过程来识别项目风险和产品质量风险。而我通常把这过程分成两个分开的过程，因为这两个过程经常有着不同的交付物和利益相关者。我将项目风险识别过程放在测试计划过程中，而质量风险识别过程并行地发生在项目早期。

这就是说，项目风险经常作为质量风险分析的副产品被识别出来，这不单单是指测试，也包括整个项目中的其他活动。另外，如果您在质量风险分析过程中使用需求规格说明、设计规格说明、用例和其他文档，可以同时发现这些文档中的缺陷，这些是质量风险分析过程中很有价值的副产品，您应当捕捉这些缺陷并且把它们上报给合适的人。

以前我鼓励大家让所有可能的利益相关方的代表参与风险管理过程。对于风险识别活动，更多利益相关者的参与将会使风险识别更加完整、准确和精确。在风险管理过程中利益相关方代表参与得越少，风险甚至风险类别越容易被遗漏。

风险识别应当做到什么粒度呢？这取决于采用的风险识别技术。在我经常使用的非正式技术中，风险识别的粒度细到风险项。每项风险项必须足够明确，使得可以对其进行分析和评估，进而得到明确的可能性级别和影响级别。

更正式的技术经常是在过程的“下游”搜索，以识别风险项一旦成真所造成的潜在影响。这些影响包括了对系统的影响或是对综合系统的影响、对潜在用户的影响、对客户的影响、对利益相关者的影响甚至对社会的影响。失效模式和影响分析是正式的风险

管理技术中的一种，它一般被应用于安全关键系统和嵌入式系统。<sup>4</sup>

另一种正式的技术在过程的“上游”搜索以识别风险的来源。危害分析就是这种正式风险管理技术的一个例子。我自己从未使用过，但我已经和客户谈论过该技术，这个客户在有严格的安全性要求的医疗系统中使用过这种技术。

我们会在本节的稍后部分看一些不同正式级别的风险分析实例。

高级大纲中把风险识别的下一步称为风险分析。我倾向于称它为风险评估，因为对我而言分析应当包括识别和评估。

不论我们称它风险分析还是风险评估，它包括了对已识别风险的研究。通常我们希望对每个风险项设定合适的风险级别并对其进行分类。

我们可以使用 ISO 9126 或者其他质量分类来组织风险项。在我看来，通常只要我们不遗忘风险项，风险项的具体类别并不是那么重要。然而，在复杂的项目和大型的组织中，风险的类别会决定由谁来处理它。在这种情况下，使用风险的分类就显得很重要了。

### 3.2.4 风险级别

风险评估或风险分析的另一部分是决定风险级别。这通常由两个关键参数决定，即问题发生的可能性和影响。问题发生的可能性主要是从技术的角度考虑，而问题发生后的影响通常主要是从商务角度考虑。然而，在一些正规化的方法中考虑 3 个因素：包括严重性、优先级和可能性，或者甚至在可能性和影响性这两个因素下继续细分。我们会在本书后续章节讨论这部分内容。

那么我们应当考虑哪些技术因素呢？下面列出供参考：

- 技术和团队的复杂度。
- 个人和培训问题。
- 团队内部和团队之间的沟通冲突。
- 供应商和客户的合同问题。
- 开发组织的地理分布以及外包。
- 新老技术方法的冲突。
- 所用工具和技术的质量。
- 管理上或技术上较差的领导力。
- 时间、资源和管理压力，尤其是与经济惩罚相关时。
- 在软件生命周期早期未引入测试和质量保证活动。
- 项目中需求、设计和代码的频繁变更。
- 高缺陷率。
- 复杂的接口和集成问题。
- 缺乏足够的需求文档。

那么我们应当考虑哪些商业因素呢？下面列出供参考：

- 受影响部分的使用频率和重要性。

---

4 关于失效模式和影响分析的讨论，参见 Stamatis 的“Failure Mode and Effect Analysis”。



- 潜在的形象损害。
- 客户和商业损失。
- 潜在的金融、生态或社会方面的损失或法律责任。
- 民事或刑事制裁。
- 失去许可证或类似损失。
- 缺少合理的变通方案。
- 缺陷的曝光度以及相关的公众负面效应。

以上所列仅仅只是个开始。

当决定风险级别时，我们可以采取定量的或定性的方法。在定量的风险分析中，可能性和影响等级都对应具体的数值。可能性用百分比表示，而影响经常用货币单位表示。如果我们把这两个值相乘就可以计算出风险发生的成本。在保险业中我们称之为预期花费或者预期损失。

虽然有朝一日如果软件工程能够经常使用定量方法会非常好，但通常来讲只能定性地分析风险级别。为什么？因为我们没有执行定量的质量风险分析所需的有效的统计数据。因此我们能够说发生的概率是很高、高、中、低或很低，但是我们很难确切地说发生的概率是 90%、75%、25%或是 10%。

这绝非是说定性的方法一定是较差的或者根本无用的。实际上，根据大部分人工作中遇到的数据来看，定量的方法在大部分的项目中都是不合适的。不准确的数据会使利益相关者对您的理解程度和风险管理程度产生错误理解。我发现如果接受数据的局限性并且应用合适的非正式质量风险管理方法，结果不仅非常有用而且对管理良好的测试流程也十分重要。

除非您的风险分析是基于广泛的统计意义上有效的风险数据，否则您的风险分析反映的是分析人员主观得到的可能性和影响。换句话说，利益相关者的个人感受和选择将会决定风险级别。我要再次强调我并非提出这些来否定定量的方法，关键是项目经理、程序员、用户、商务分析师、架构师和测试人员通常有不同的感知，从而可能对每个风险项的风险级别有不同的观点。通过让所有人参与，我们可以获得团队智慧的精华。

然而，利益相关者之间也很有可能无法达成一致。所以风险分析过程中应当包括达成一致的方法。最起码在所有人无法达成一致时，可以把不一致的内容上报到更高一级的管理层解决。否则，风险级别将变得模糊不清和相互冲突，从而无法指导包括测试在内的风险缓解活动。

### 3.2.5 控制风险

包括测试经理在内的任何管理人员都在控制各自领域内的风险。我们如何控制风险呢？主要有 4 种风险控制方法：

- 风险缓解，采用预防手段降低风险的可能性和/或影响。
- 应急方案，通过一个或多个计划来降低风险一旦发生后的影响。
- 风险转移，将风险转移给另一方承担。
- 最后，我们可以忽视或接受风险和它所产生的后果。



对于任何风险项,选择以上一种或多种风险控制方法都会带来不同的收益和机会,也包括随之而来的成本和潜在的额外风险。选择错误的风险控制方法会使情况变得更糟,而不会更好。

分析式的基于风险的测试专注于为测试团队创造风险缓解的机会,特别是针对质量风险。基于风险的测试通过贯穿整个软件生命周期的测试缓解质量风险。

在某些情况下我们可以应用一些标准。本章会简单介绍一些风险相关的标准。

### 3.2.6 项目风险

虽然本章大部分内容讨论产品风险,但是测试经理经常识别项目风险并且有时他们也不得不管理这些项目风险。让我们在这一节讨论项目风险,然后可以在下一节集中讨论产品风险。具体的所有可能与测试相关的项目风险清单会很长,但主要包括以下类似问题:

- 测试环境和工具就绪状态。
- 测试人员是否到位、是否胜任。
- 低质量的测试交付物。
- 范围或产品定义频繁变化。
- 低质量和随机的测试。

测试相关的项目风险经常能够被缓解,或者至少在它不幸发生以后有一种或多种应急方案对它采取措施。测试经理可以用许多方法管理项目风险。

我们可以让测试更早地介入从而保证更早完成测试件的准备。这样我们能够保证当产品准备就绪后我们就可以开始测试。另外,就像在初级大纲和本书其他部分提到的一样,测试团队在早期介入项目可以使测试分析、设计和实现作为项目静态测试的一种手段,从而防止缺陷在诸如系统测试的动态测试晚期才被发现。在高级别测试诸如系统测试、系统集成测试和验收测试中发现大量意想不到的缺陷会产生严重的项目延期风险,因此,上面提到的缺陷预防活动是测试活动带来的关键收益,因为它降低了项目风险。

我们可以保证在测试执行开始之前检查测试环境。这项检查伴随着另一项风险缓解活动,那就是在正式的测试执行开始前测试早期版本。如果我们在测试环境中这么做,那么我们就能够在测试正式开始前测试测试件、测试环境、测试发布、测试对象安装过程和许多其他测试执行过程。

我们也可以定义更严格的入口准则。如果项目经理在项目延期开始时也会延期结束项目,那么这会是很有效的方法,但项目经理们通常不会这么做。因此在不改变项目测试结束时间的情况下,开发活动的延期只会给测试活动带来更大的压力。

我们可以尝试改进系统的可测试性需求。例如在可能的情况下让用户界面团队把可修改输入框改成不可修改的下拉菜单,比如日期和时间。这样可以大量地减少可能的用户输入确认测试工作量并有助于自动化。

为了降低不良测试对象不期而至的可能性,同时也帮助减少这些测试对象中的缺陷,测试团队成员可以更早地参与项目工作的评审,例如需求规格说明评审。也可以让测试团队参与到问题和变更管理。

最后，在测试执行中我们可以监测项目进度和质量。这最好能够从单元测试或更早开始，但最迟不要晚于正式测试开始的那一天。如果看到情况越来越糟，我们可以在他们搞砸项目前尝试管理他们。

在图 3-1 中可以看到一个与测试相关的项目风险的例子，该项目是一个 Internet 应用项目，这个例子在本书中会反复使用。这些风险在测试计划中被识别后，我们通过风险缓解或应急方案，在项目中分步骤地对这些风险进行管理。

风 险	缓解/应急措施
不能按时组建测试团队	按照优先级的倒序减少测试的范围
没有很好地定义发布管理，导致一个测试周期的结果无效	定义良好的发布管理流程
缺乏测试环境系统管理员的支持	获得系统管理员的寻呼机/手机，并保证系统管理员具备相关技能
测试和开发共享环境	[待决定]
开发交付物的质量太差影响测试进度	完善的单元测试。坚持测试入口和出口准则。尽早审计供应商的测试、可靠性计划和结果
测试以及产品的范围和定义的变更影响测试进度	变更管理或变更控制委员会

图 3-1 测试相关的项目风险举例

让我们回顾一下该项目测试中识别到的主要项目风险以及为这些项目风险所做的风险缓解和应急方案。

考虑到起初激进的进度计划，我们担心测试团队成员可能无法准时到位。我们的应急方案是按照优先级的倒序减少测试人力投入。

在一些项目里，测试发布管理并没有被很好地定义。这可能导致一个测试周期的结果作废。我们的缓解方案是建立一个明确定义且简明扼要的发布管理过程。

有时我们在关键时刻会遇到测试环境不可用或无法在测试环境中开展工作的情况，这需要系统管理员的帮助。我们的缓解方案是识别系统管理人员，登录他们的 BP 机，手机联系方式以及相应的 UNIX、QNX 和网络管理技能。

作为咨询师，我和我的同事经常遇到和开发人员共享测试环境的情况。这会造成测试执行进度极大的延误和无法预料的中断。在这种情况下，我们还没有找到最佳的缓解或应急方案，因此它被标记为[待决定]。

当然，有缺陷的交付物会妨碍测试过程。一个新应用（与维护发布相对比）的测试周期长短，实际上，多数情况下，取决于产品中缺陷的数量以及多久能修复这些缺陷。我们把完整的单元测试以及坚持测试入口和出口准则作为针对新应用的风险缓解计划。对于硬件部分，我们可以通过对供货商测试、可靠性计划和结果的早期审查来缓解风险。

还有一种情况是测试、产品范围和产品定义频繁的变更阻碍测试过程。我们应对这种情况的应急方案是建立一个变更管理或变更控制委员会。



### 3.2.7 两种工业标准以及它们与风险的关系

我们在高级大纲中提到过 ISO/IEC 61508, 您可以在这个标准中找到一个关于如何在复杂的和/或有严格安全要求的系统工程中进行风险管理的有趣的例子, 这包括质量风险管理。它是针对有安全要求的控制系统中的嵌入式软件而设计的, 这一点您可以从它的标题中看出来: “安全相关的电子系统/电子设备/可编程电子系统的功能安全性”。

该标准关注风险, 因此需要进行风险分析。它包括决定风险级别的两个主要因素: 可能性和影响。该标准指导我们通过改进系统中的电子应用、电子设备或软件, 把项目中的剩余风险级别降低到可接受的水平。

这个标准有一个关于风险的固有哲学。它承认无论是整个系统还是单个的风险项, 我们都不可能达到零风险。因此我们不得不在开始的时候而不是在后期就关注质量, 特别是安全性。必须采取类似需求、设计和代码评审这样的缺陷预防措施。

同时这个标准也坚持要求我们了解哪些是可接受的风险, 哪些是不可接受的风险。我们应当采取措施来减少不可接受的风险。当这些措施与测试相关时, 我们必须把它们文档化。这包括了软件安全性确认计划、软件测试规格说明、软件测试结果、软件安全性确认、验证报告和软件功能安全性报告。这个标准还涉及作者偏袒的问题, 也就是关于自己测试的问题。您应当还记得在基础级大纲中我们曾提过这个问题, 因此我们要求进行独立的测试, 尤其是针对那些安全性要求严格的测试。此外, 因为当系统具有可测性时进行测试是最有效的, 所以系统可测性也是一种需求。

这个标准有一个关于安全性集成级别 (SIL) 的概念。SIL 基于在特定的组件或者子系统中失效的可能性, 它会影响许多风险相关的决定, 其中包括测试和质量保证技术的选择。

有些技术是我在高级测试分析师一卷中所提到的, 例如各种功能测试和黑盒测试设计技术。许多技术是我在高级测试技术分析师一卷中所提到的, 包括随机测试、动态分析、数据记录和分析、性能测试、接口测试、静态测试以及复杂度标准。另外, 由于要达到完全的覆盖率必须降低缺陷遗漏率, 所以这个标准要求使用合适的自动化测试工具。

根据安全性集成级别, 这个标准需要各种级别的测试。这包括组件测试、集成测试、软硬件集成测试、安全性需求测试和系统测试。如果需要某种级别的测试, 这个标准要求文档化及独立验证。换句话说, 这个标准可以要求对测试活动进行审计或外部评审。顺着“保卫守护者”这个思路, 这个标准也要求对测试用例、测试规程和测试结果进行评审, 同时根据测试条件确认数据完整性。

ISO/IEC 61508 要求把结构化测试作为测试设计技术。这也就间接地要求根据安全性集成级别实现结构化覆盖。因为希望对安全关键系统有更强的信心, 这个标准要求有完整的需求覆盖率, 并且不是在一个级别上而是在多个测试级别上。所需的测试覆盖率级别依赖于安全性集成级别。

现在这标准看上去有些不合时宜了, 特别是如果当您来自于一个充满了非正式活动的世界。然而, 当下次您走到两块可移动的金属中间的时候, 比如电梯门, 问一下自己想要在这样的控制系统软件中保留多少风险。



让我们看一下另一个风险相关的测试标准。美国联邦航空局为航空系统制定了一个被称为 DO-178B 的标准。在欧洲，这个标准被称为 ED-12B。

如表 3-1 所示，DO-178B 标准按潜在的失效影响定义了危急程度。根据危急程度，DO-178B 标准要求特定的白盒测试覆盖率级别。

表 3-1 FAA-DO 178B 标准规定覆盖率

危急程度	潜在的失效影响	所需的结构覆盖
级别 A：灾难性的	软件失效可能导致灾难性的系统失效	条件判定、判定和语句覆盖
级别 B：危险的/严重的	软件失效可能造成危险的或严重/主要的系统失效	判定和语句覆盖
级别 C：主要的	软件失效可能造成主要的系统失效	语句覆盖
级别 D：轻微的	软件失效可能造成轻微的系统失效	无
级别 E：无影响	软件失效不会对系统造成影响	无

- 危急程度 A——灾难性的，软件失效可能导致灾难性的系统失效。对于这种危急程度的软件，DO-178B 标准要求条件判定、判定和语句覆盖。
- 危急程度 B——危险或严重的，软件失效可能造成危险的或严重/主要的系统失效。对于这种危急程度的软件，DO-178B 标准要求判定和语句覆盖。
- 危急程度 C——主要的，软件失效可能造成主要的系统失效。对于这种危急程度的软件，DO-178B 标准只要求语句覆盖。
- 危急程度 D——轻微的，软件失效可能造成轻微的系统失效。对于这种危急程度的软件，DO-178B 标准不要求达到任何级别的覆盖。
- 危急程度 E——无影响，软件失效不会对系统造成影响。对于这种危急程度的软件，DO-178B 标准不要求达到任何级别的覆盖。

这个标准很有意义。您应当更关注影响飞行安全性的软件，例如方向舵和副翼控制模块，而不是类似视频娱乐系统那样的其他软件。当然，后来将关键和非关键软件都构建在同一飞机的网络上成为一种趋势，这引入了巨大的潜在风险，比如未知干扰和恶意攻击。

然而，我认为只用白盒测试相关的度量来决定我们对系统的信心仍然是危险的。覆盖率度量元确实是一种保证信心的手段，但是我们应当同时运用包括黑盒和白盒在内的多种覆盖率度量元手段。<sup>5</sup>

顺便提一句，如果您觉得这部分内容理解有困难，注意在这个标准中所用的白盒覆盖率度量在基础级大纲第 4 章中有讨论，您可以回到基础级大纲第 4 章复习相关的内容。

### 3.2.8 基于风险的测试与失效模式和影响分析练习 1

回顾一下附录 B 中 HELLOCARMS 系统需求规格说明，寻找其中的项目风险，特别

<sup>5</sup> 您可能会说，“为什么我要担心这些？看上去这方法对航空软件来说已经很好了。”花点时间看一下 [www.risks.org](http://www.risks.org) 网站上的风险摘要并且仔细阅读那些与软件相关的几乎导致航空事故的部分，您可能就不会这么乐观了。其中有人对波音 787 设计中安全关键系统和非安全关键系统都使用同一网络这个问题进行了讨论。

注意识别测试相关的项目风险。

把测试相关的项目风险文档化并且记录下针对这些风险所选择的控制方法。

如果可能的话, 请进行小组讨论。

### 3.2.9 基于风险的测试与失效模式和影响分析练习 1 参考答案

下面是我通过评审 HELLOCARMS 系统需求规格说明所识别的与测试相关的项目风险以及其他的项目风险。对于与测试相关的项目风险, 我已经识别了相应的控制活动。

#### 3.2.9.1. 项目测试风险

下面是需求中的一些与测试相关的项目风险以及对这些项目风险的可选的控制活动。

1. 无法保证复制真实的生产环境(参见附录 B 章节 000)以供测试, 这可能导致测试受限甚至停滞。

- 风险缓解: 识别在测试中需要复制生产环境的质量风险, 将其报告给项目经理。
- 风险缓解: 当测试环境构建完成, 识别出由于测试环境和生产环境不一致而无法进行的测试, 将其报告给项目经理。
- 风险缓解: 研究性能测试和可靠性测试外包的可能性, 因为其应用架构是通用的。
- 风险转移: 如果在测试执行过程中仍然存在测试环境不一致的问题, 将该不一致引起的测试覆盖率问题上报给整个项目团队。
- 应急方案: 如果特定的交互系统不可用(例如评分系统), 可以安排测试用具来模拟这些交互操作。

2. 无法获得和部署足够的、可工作的性能测试和可靠性测试工具(参见附录 B 章节 020 和 040)。

- 风险缓解: 在测试计划时确定预算和其他限制条件, 然后在工具评估中应用这些限制条件。
- 风险缓解: 在测试计划的同时开始工具评估。
- 风险转移: 如果因为缺少工具的原因导致性能测试和可靠性测试无法进行, 将这个测试覆盖率受限的问题上报给项目团队。

3. 在正式测试开始时因为测试发布的质量(包括可卸载性)不达标而无法运行测试, 包括基本功能性测试和互操作性测试。

- 风险缓解: 使用迭代周期相对较短的迭代生命周期模型, 致力于在第一个迭代周期里达到基本功能性和互操作性。
- 风险缓解: 持续集成、每日构建以及开发团队执行自动化的冒烟测试。
- 风险缓解: 定期地在测试实施阶段对测试发布安装进行测试(例如在测试执行前), 从而在测试执行开始前评估测试环境和测试发布的可安装性/可测性。
- 风险缓解: 设计同时覆盖端对端、复杂工作流和相对简单能力(例如: 分屏)的测试, 使得测试在面对低质量软件时仍能保持健壮性。
- 应急方案/风险转移: 如果测试无法正常开始, 那么可以根据风险优先级选择测试用例并将测试覆盖率降低的问题上报给项目团队。



4. 在第二个迭代周期后发生的产品变更需要额外的再测试、原测试结果的废除和测试更新。

- 风险缓解：在新的迭代周期里使用可维护的自动化测试对已有功能进行回归测试，在每一个迭代周期中增加新的自动化测试用于回归测试（注意这个风险缓解计划不仅包括自动化测试设计还包括安排相关的测试团队人力资源）。
- 风险缓解：使用具备领域和测试方面专家技能的测试人员进行手工测试，限制测试用例文档化的程度从而减少测试更新工作。维护重要的（10%~20%）探索性测试执行，在测试执行后采用测试会话方式记录探索性测试的覆盖率。
- 应急方案/风险转移：不可能进行回归的部分，根据风险选择测试并将测试覆盖率降低的问题上报给项目团队。

5. 需求规格说明（例如章节 020）中的空白点完成得太晚以至于无法做足够的测试准备。

- 风险缓解：从第一个测试执行周期的开始日期倒推得出所有关键规格说明必须完成的时间，并和项目经理一起确保所有关键规格说明在此日期之前完成，从而避免其影响测试。
- 风险缓解：对当前不明确的需求做出合理的假设，定义轻量级的测试，保留在测试执行过程中进行测试更新的可能性。
- 风险缓解：指定一个测试团队成员监督需求的变更/完成，从而可以针对需求变更/维护识别合适的测试用例。
- 风险转移：因为信息不够而无法进行测试的部分，将测试覆盖率降低的问题上报给项目团队。

6. 复杂测试环境中的错误配置导致大量的假事件报告（例如与环境相关但与软件本身不相关的事件报告）。

- 风险缓解：指派具备足够技能的可用的系统管理员。
- 风险缓解：定期地在测试实施阶段对测试发布安装进行测试（例如在测试执行前），从而在测试执行开始前评估测试环境和测试发布的可安装性/可测性。[注意：与前面第3项中的风险缓解内容重复]。
- 应急方案：监督测试执行中与环境相关的假事件报告，必要时聘请合同工或咨询师来增强测试系统管理员的技能。

7. 测试覆盖率的不足导致缺陷发现率无法接受（例如低于 95%），从而导致现场缺陷率高。

- 风险缓解：运用质量风险分析来指导测试。
- 应急方案：运用重要的（10%~20%）探索性测试执行组合（包括非关键风险部分），运用探索性测试结果检查风险分析和测试用例中的不足，根据发现的不足重构测试。
- 风险转移：把与关键风险（高可能性和/或影响）相关的测试不足上报给管理层。

8. 测试执行开始时间延期，但测试执行结束时间不变。这导致测试时间被压缩。

- 风险缓解：坚持使用迭代生命周期模型，不通过减少测试范围来压缩迭代中测试

的时间，而是通过减少迭代周期的数量从而减少测试量。

- 应急方案/风险转移：当测试必须被压缩时，根据风险选择测试并将测试覆盖率降低的问题上报给项目团队。

9. 预算或人员不足导致无法构建完全合适的测试团队和/或环境。

- 风险缓解：研究通过外包降低成本的可能性（例如性能测试、可靠性测试和兼容性测试）。

- 风险缓解：将测试工作结构化从而使那些分离的、可拆分的测试部分可以被剥离，进而在现实的时间框架内尽可能做有用的测试。

- 应急方案/风险转移：当测试必须被压缩时，根据风险选择测试并将测试覆盖率降低的问题上报给项目团队。

10. 在测试环境中调试引发不可预测的延误和测试时间/成果损失，最终导致测试团队的测试被压缩。

- 风险缓解：和项目经理一起保证给开发团队提供足够的调试环境（这同时也可以降低可安装性问题的风险，因为软件会首先在开发环境中被安装）。

- 应急方案/风险转移：当测试必须被压缩时，根据风险选择测试并将测试覆盖率降低的问题上报给项目团队。

### 3.2.9.2 其他项目风险

下面是需求中的一些与测试无关的项目风险。

1. 考虑到安全需求不清楚（参见 010-040-010），支撑项目进度的必要的基础结构很可能无法完成。

2. 如果给电话银行客户经理的上报数量超过预期（参见附录 B 章节 001 和 004）可能会使系统无法按预期的客户量提供服务。

3. 缺乏开发人员或是缺乏分配到其他交互系统（例如评分系统、贷款文件打印系统等）来解决跨系统的问题的 Globobank 雇员，这会导致缺陷修复、生产环境配置和第一次发布延迟。

4. 相关系统的变更/升级未与该版本协调一致。

## 3.2.10 风险识别和评估技术

当今有许多质量风险识别和评估技术。这些技术分为非正式、半正式和正式。

您可以把风险识别和评估活动想象为项目和产品评审的结构化形式。在需求评审时我们主要关心系统应当做什么，而在质量风险识别和评估中，我们主要关心那些系统不应当做却可能会做的部分。因此，我们可以把质量风险识别和评估活动看成是需求、设计和实施的镜像。

随着评审正式级别的提高，缺陷移除有效性、文档化程度和成本也随之提高。你将根据项目的需要和限制来选择技术。例如，如果你工作在一个预算非常有限的小项目上，采用需要大量文档化工作的正式的技术是不合适的。

让我们回顾一下从非正式到正式的各种质量风险识别和评估技术以及可以用来组织评审的一些方法。



在许多成功的项目实施中，我们使用了非正式的、基于风险的测试方法。这些方法工作得很好。它特别适合学习和练习基于风险的测试，因为过多的正式性和文档工作可能会对基于风险测试的成功实施造成障碍。

运用非正式技术时，我们主要依靠历史数据、利益相关者领域和技术方面的经验和风险类别检查表。这些非正式方法很容易实施和执行，它们无论在文档方面还是在时间投入方面都是轻量级的。在项目切换时这些方法很灵活，因为文档化的过程数量已经被减到最少。

然而，正因为我们如此依赖于利益相关者的经验，这些技术和参与者紧密相关。错误的参与者会导致风险项和评估出的风险级别准确性降低。因为我们遵循检查表，如果检查表存在不足，那么我们的风险分析也会存在不足。因为风险项被识别在相对高的级别，所以就风险项和其相应的风险级别而言，它们可能不够精确。

那就是说，这些非正式技术是进行基于风险测试很好的开始。如果证实需要更精确或更正式的技术，可以在随后的项目里将非正式质量风险分析方法扩展和正式化。即使有经验的基于风险测试的使用者也应该考虑在低风险或敏捷项目中应用非正式技术。应当避免在安全关键项目或规定项目中使用非正式技术，因为它们精确度较差，并可能和实际情况存在一定的偏差。

### 3.2.11 质量风险分类

我提到非正式的基于风险的测试倾向于依赖检查表来识别风险项。那么风险分类有哪些呢？在某种程度上，这决定于我们考虑的测试级别。让我们从早期的测试级别，单元或组件测试开始。在接下来的列表里，我将会用提问的形式列出检查表列表，以此来模拟你的思考方式。

- 状态：单元是否恰当地处理状态相关的行为？当事件发生时状态转换是否正确？行动是否正确触发？是否每个输入都正确地关联到事件？
  - 业务：单元是否正确地处理业务？是否会产生不希望有的副作用？
  - 代码覆盖率：哪些语句、分支、条件、条件组合、循环以及其他代码路径可能导致失效？
  - 数据流覆盖率：哪些数据流出入单元可能导致立即失效或延时失效，包括持久数据的损毁（最差的数据毁坏类型）？是否采用参数、对象、全局变量，或比如文件或数据库表格这样的持久性数据结构？
  - 功能：这个组件实现的系统功能是否正确？是否有负面影响？
  - 用户接口：如果这个组件会和用户互动，用户是否能够正确理解提示信息？用户是否会对配色方案和图片满意？
  - 机械寿命：硬件组件是否可能在反复运转和使用后损坏？
  - 信号质量：硬件组件信号是否正确以及信号格式是否正确？
- 随着我们进入集成测试，许多新的风险在以下部分凸显出来：
- 组件或子系统接口：组件间接口是否已经被定义清楚？哪些问题可能在组件直接或间接相互作用时发生？

- 功能: 在各个不同的动作和副作用中可能存在哪些问题? 特别是当作为组件间交互的结果时。
- 容量: 静态数据空间, 例如内存和硬盘空间是否足够保存所需信息? 动态通道容量, 例如网络容量是否能够提供足够带宽?
- 错误/灾难处理和恢复: 集成组件是否能在一般和极端恶劣条件下正确响应? 组件是否能在一般和恶劣条件下恢复正常功能使用?
- 数据质量: 系统是否能可靠地进行数据存储、装载、修改、归档和操作而不会出现数据损坏或丢失?
- 性能: 就响应时间、资源有效使用率等参数而言可能存在哪些问题?
- 用户接口: 对于集成组件, 如果含有用户接口, 用户是否能够正确理解提示信息? 用户是否会对配色方案和图片满意?

类似的问题也出现在系统集成测试中, 不同的是我们关心的是系统集成而不是组件。最后, 在系统和用户验收测试中有哪些风险呢?

- 功能: 我们重申需要考虑功能问题。在这里我们关心的是系统级别的问题。端对端功能是否正确工作? 深层次的功能是否工作? 各种功能能否协同工作?
- 用户接口和可用性: 整个系统的用户接口是否一致? 用户是否能够理解所提供的用户接口? 我们是否在某一点误导用户或使用户感到困惑? 是否存在没有出路的用户接口?
- 状态和业务: 总地来说, 系统是否正确地处理各种语句? 考虑到系统中用户或对象的语句, 是否可能存在问题?
- 数据质量: 考虑到整个系统使用的所有数据, 包括可能和其他系统共享的数据, 系统是否能可靠地进行数据存储、装载、修改、归档和操作而不会出现数据损坏或丢失?
- 运行: 复杂的系统经常需要系统管理工作, 例如, 管理数据库、网络和服务。这些系统管理工作会带来大量的维护工作。例如, 备份和恢复文件或数据库是否可能有问题? 是否能对系统进行数据库版本升级、不同数据库间迁移或不同中间件间迁移? 是否能够增加硬盘空间、内存或提高中央处理器处理能力?
- 性能、压力和容量: 响应时间是否有问题? 高压低资源下是否会引发问题? 静态空间不够的情况下是否会引发问题? 动态通道的容量和带宽不足是否会引发问题?
- 可靠性、可用性和稳定性: 系统是否会在正常的、异常的和高压情况下无法工作? 系统是否可能在需要使用时不可用? 是否有某些功能不稳定?
- 安装、切换、设定和配置: 哪些安装、数据迁移、应用迁移、配置或初始条件可能引发问题?
- 错误/灾难处理和恢复: 系统是否能在一般和极端恶劣条件下正确响应? 系统是否能在一般和恶劣条件下恢复正常功能使用? 系统遇到恶劣情况做出反应后, 是否会对同一环境里的其他应用产生不良影响?
- 日期和时间处理: 是否在特定的日期和时间会导致事件失败? 使用日期或时间的



相关功能是否能正确地协同工作？闰年和夏令时调整是否可能引发问题？时区变化是否可能引发问题？

- 本地化：就我们需要支持的各种语言而言，是否存在一些字符集或翻译信息可能引发问题？货币单位不同是否可能引发问题？
- 网络化和分布式：等待时间、带宽或其他与网络或分布式处理和存储有关的因素是否可能引发潜在的问题？
- 兼容性：系统是否可能会和相关的环境存在兼容性问题？系统是否可能会与同一环境中的其他应用存在兼容性问题？
- 标准：系统遵照哪些标准？系统是否可能违反这些标准？
- 安全性：用户是否可能访问到他们权限以外的功能和数据？用户是否可能会无法访问到他们权限内的功能和数据？数据是否在必要的情况下进行了加密？是否可以绕过权限控制进行安全性攻击？
- 环境：在正常和异常的运行环境下是否会出现硬件系统失效？潮湿、灰尘或高温是否会导致间歇性或永久性的失效？
- 电源：硬件系统是否具有电源消耗相关问题？常规的电源供电量变化是否会引起问题？电池寿命是否足够？
- 冲撞、振动和跌落：对于硬件系统，对于可预见的物理冲撞、后台振动或常规颠簸和跌落是否会导致失效？
- 文档和打包：文档是否正确、充分和有帮助？打包是否充分？
- 可维护性：系统能升级吗？能打补丁吗？可以在安装后卸载已有功能或安装新功能吗？
- 当然还有其他潜在的风险类别，但以上所列是一个很好的起点。在使用这些风险分类列表时，你可以根据系统的实际情况进行定制。

### 3.2.12 记录质量风险

在图 3-2 中可以看到一个用来填写质量风险分析信息的模板。在这个模板里，我们首先使用上一节讨论过的风险分类来识别风险项。然后，根据可能性和影响评估它们的风险级别。接下来用可能性和影响来决定总体的测试优先级和测试程度。最后，如果风险存在于特定的需求或设计规格说明中，那么建立风险项与这些文档的关联。让我们看一下这些活动以及它们如何使用这个模板来生成信息。

首先记住质量风险是可能降低用户满意度的潜在系统问题。我们可以使用前面的风险分类来组织列表并提醒大家相关的风险项，然后和利益相关者一起在每个分类中识别一个或多个风险项。

得到已识别的风险后，我们可以检查风险项列表并且评估风险级别，因为我们能够看到风险项相互间联系。非正式技术通常使用主要因素来评估风险。

首先是问题的发生可能性，这主要是从技术角度考虑，有时我称它为“技术风险”。其次是问题造成的影响，这主要是从商务或运行角度考虑，有时我称它为“商务风险”。

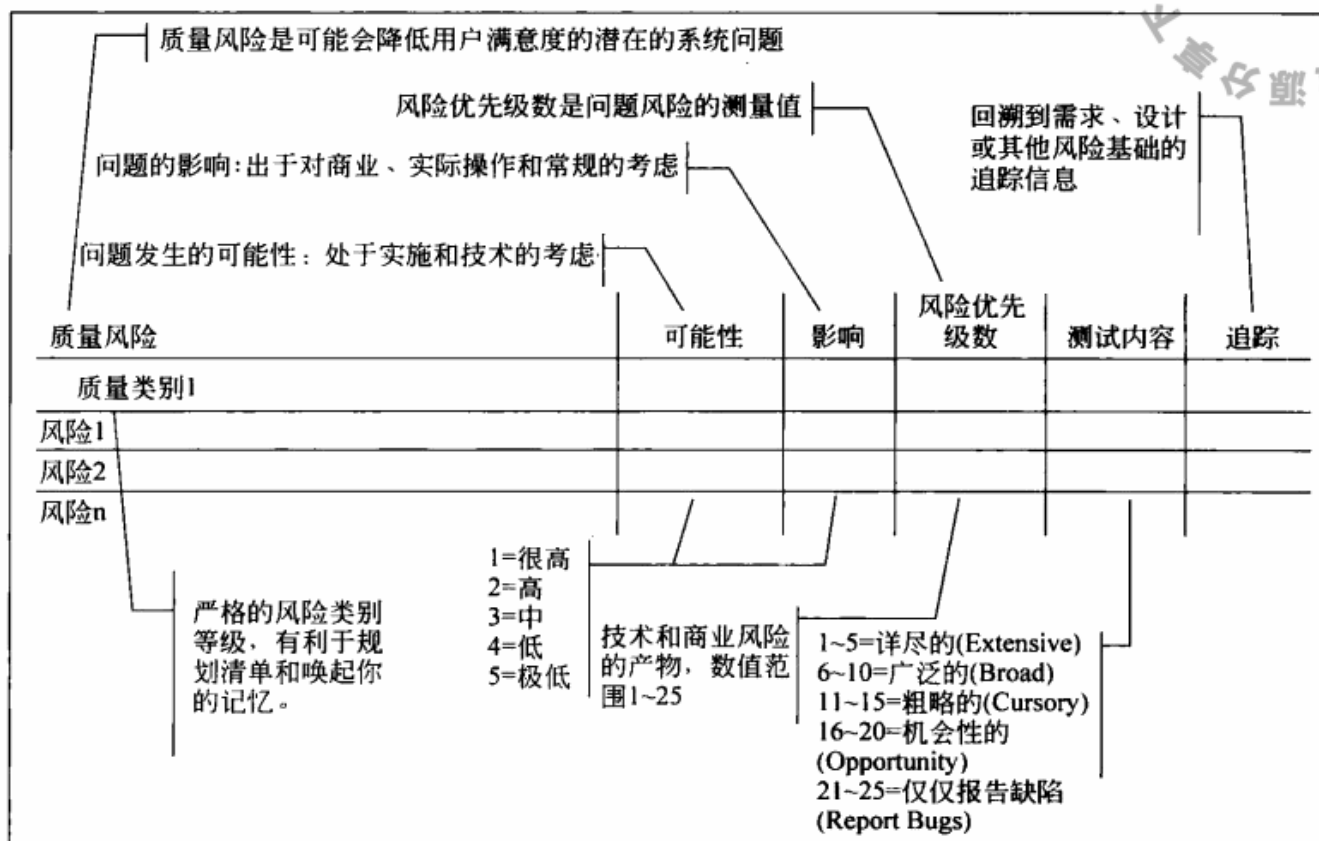


图 3-2 质量风险信息模板

可能性和影响都可以通过等级分类来评分。三分制的等级分类可以划分为高、中和低。我倾向于使用五分制的等级分类，评分可以划分为很高、高、中、低和很低。

考虑到可能性和影响，我们可以对质量风险项进行单个的、合计的风险度量计算。一般我们称这种风险度量为风险优先级数字。一种方法是使用公式通过可能性和影响计算风险优先级数字。首先，把等级分类转换为数字的级别，例如：

- 1 = 很高
- 2 = 高
- 3 = 中
- 4 = 低
- 5 = 很低

然后可以用可能性和影响相对应的数字计算出风险优先级数字。我们将会在本章的后续章节中重新讨论这个问题，因为这只是许多决定风险优先级方法中的一种。

风险优先级数字可以被用来决定测试顺序。要分配测试工作必须决定测试范围，图 3-2 展示了一种通过将优先级分成 5 组的方法来分配测试工作：

- 1~5 = 详尽的测试
- 6~10 = 广泛的测试
- 11~15 = 粗略的测试
- 16~20 = 机会性的测试
- 21~25 = 仅仅报告缺陷



我们将会在本章后续段落中继续讨论实现以上目标的其他各种方法。

正如前面提到的那样，当你进行质量风险分析时，可能会同时生成各种有用的副产品。这包括你和利益相关者在评估问题可能性时所提出的系统实施假设。你会想要去验证这些假设而它们的确可能会被证明是有用的建议。有用的副产品还包括你发现的项目风险。此外，最重要的是副产品中包括了需求、设计或其他文档中的问题。我们可以避免将这些问题变成真正的系统问题。注意本书已经在前面的章节中讨论过所有这3种缺陷预防作用。

图3-3是一个非正式质量风险分析实例。我们主要使用了6种质量风险分类：

- 功能。
- 本地化。
- 可用性和用户接口。
- 可靠性。
- 性能。
- 可支持性。

这些是HP公司的一些人使用的标准质量分类。

我已经针对每种风险分类举了一到两个质量风险的实例。当然对于产品而言通常总共会有100~500个质量风险，对于特别复杂的产品甚至可能会更多。

### 3.2.13 用ISO 9126进行质量风险分析

如果愿意，我们可以使用ISO 9126标准来替代在前面提到的相当冗长且非结构化的质量风险分类列表，从而让非正式风险分析稍微正式化，也使它变得更加结构化。

ISO 9126标准有它的一些优点，它提供了一个预定义的和考虑周到的框架。这个标准包括一整套文档，它提供了一个预定义的方法来使得它符合组织的实际需要。如果您在所有项目里都使用ISO 9126标准，您的质量风险分析以及得出的测试覆盖率都会基于一个共同的基础。项目测试中的合规性保证了结果的可比较性。

在风险分析中使用ISO 9126也有缺点。首先，如果在选择质量特性时不小心，您会发现风险分析可能涵盖范围过于广泛，这会降低工作效率。另外，项目的规模和复杂度各不相同，如果在所有项目中都使用ISO 9126标准可能会使过程控制过严和过于重量级。

当项目需要更多的正式化或结构化时以及在规格标准重要时，建议您考虑使用ISO 9126进行风险分析。我会避免在非典型的项目或那些过多的结构化、过程化或文档工作可能会引发问题的项目上使用ISO 9126标准。这些项目更适合使用轻量级的非正式的过程，如图3-3所示。

回顾一下ISO 9126标准，有6种质量特性：

- 功能，包括适用性、准确性、互操作性、安全性和与法规及标准的一致性。
- 可靠性，包括成熟度（健壮性）、故障容限、可恢复性和与法规及标准的合规性。
- 可用性，包括可理解性、易学性、可操作性、吸引性和与法规及标准的合规性。

质 量 风 险	技术风险	商务风险	风险优先级	测试内容
功能				
用户登录和认证失败	2	1	2	详尽的
用户不能查看账户事务	3	1	3	详尽的
本地化				
西班牙语翻译和相关字段的要求不符	2	3	6	广泛的
西班牙语的消息翻译错误	4	3	12	粗略的
可用性和用户接口				
在屏幕上，用户操作出现“卡壳”，尤其是出现错误的情况	4	2	8	广泛的
用户发现接口太复杂	4	1	4	详尽的
可靠性				
从 4:00 到第二天 2:00 的可用性低于 95%	4	2	8	广泛的
性能				
在完全负载情况下，系统响应时间大于 0.5 秒	5	2	10	广泛的
可支持性				
补丁和发布无法正确安装	5	4	20	机会性的
补丁和发布需要的宕机时间太长	5	3	15	粗略的

图 3-3 非正式质量风险分析实例

- 效率，包括时间行为、资源使用和与法规标准的合规性。
- 可维护性，包括可分析性、可变性、稳定性、可测性和与法规及标准的合规性。
- 可移植性，包括适应性、可安装性、共存性、可替换性和与法规及标准的合规性。

同时应当牢记的是在 ISTQB 黑盒测试或行为测试分类中，那些和功能以及它们的子特性相关的测试是功能测试。而那些和可靠性、可用性、效率、可维护性和它们的子特性相关的测试则是非功能测试。

3.2.14 用风险发生成本进行质量风险分析

另一种质量风险分析形式是通过风险发生成本来进行分析。风险发生成本这个名字来源于金融和保险业。风险发生成本在保险业中被称为预期花费，它是由损失的可能性和损失的平均成本相乘得出的。通过长期的和大量的风险抽样研究，我们认为损失总量趋向于接近所有风险的发生成本。

因此，对于每个风险我们都应当进行评估和成本分析来决定是否进行测试。如果测试成本低于风险发生成本，我们应当对该风险进行测试从而获益。如果测试成本高于风险发生成本，那么测试就并非是降低该风险成本的好办法。

这显然是一种非常明智和平衡的测试方法。有收益的话我们就进行测试，反之我们就不进行测试。还有什么比这更具有实践性呢？在保险和金融业里，你会发现利益相关者很好地把这点和这种方法联系在一起。



但这也就是说，这种方法有一些它的问题。为了放心地使用这种方法，我们需要足够的数据从而能对可能性和成本做出合理的估计。而且，这种方法主要根据金钱的数量来决定测试范围和顺序。而对于有些风险而言，主要的影响并非金钱方面的或至少很难用金钱进行量化，例如失去一笔生意或企业形象受损。

如果我正工作在一个金融或保险业的项目上并且可以获得相关数据，那么我很可能使用这种方法。在风险分析过程中技术对其他参与者的可访问性是相当有价值的。然而，我会避免在安全关键或业务关键的项目里使用这种技术。人身伤害或对生意产生灾难性影响这类风险是无法用金钱来正确衡量的。

### 3.2.15 用危害分析进行质量风险分析

另一种可以使用的风险分析技术被称为危害分析。和风险发生成本分析一样，这种分析技术对某些领域非常适合但并非适用于所有领域。

危害是会生成风险的事或物。例如湿滑的浴室地板有使人滑倒摔伤的风险。在危害分析里，我们尝试理解这些会使我们系统产生风险的危害。在测试以及上游活动中运用这种技术可以减少危害从而降低风险发生的可能性。

如您想象的那样，这是一种非常严谨的、审慎的和系统的技术。当识别风险以后，我们必须思考风险是如何形成的以及如何处理那些产生风险的危害。在我们无法承受任何风险遗漏的情况下，这种技术很有意义。

但是，在复杂的系统里有成百上千的危害相互作用产生风险，而且许多危害是我们无法预测的。因此，危害分析不适合复杂度过高的情况。在这种情况下，我们通过危害分析认识到的风险会少于实际的风险数量，这不是我们所期望的。

我会考虑在医药或嵌入式系统项目里使用危害分析技术。但是，我不会在无法预测的、快速演变的或高度复杂的项目里使用这种技术。

### 3.2.16 判定所有风险的优先级

一会儿会讨论另一种风险分析方法，但我想先介绍另一个问题。根据风险的各种因素计算得出综合风险优先级。我们在前面用风险可能性乘以影响得出风险优先级数字时已经提到过这种技术。这种技术也用于风险发生成本技术，风险发生成本可以通过可能性和风险的平均损失成本相乘得到。

有些人更喜欢使用加法而非乘法。例如 Rick Craig 用可能性和影响进行加和计算<sup>6</sup>，这样会使风险规模更加扁平 and 集中。为了更好地解释，让我们花一些时间构建两个表。将可能性和影响分别分成 1~5 的等级，用可能性和影响的所有组合计算得到所有可能的风险优先级数字，然后将其填入表格。每种表格都有 25 个单元格。对于加法计算风险优先级数字的范围是 2~10，而对于乘法计算风险优先级数字的范围是 1~25。

也可以通过构建复杂的公式来计算风险优先级数字，其中有些方法是通过使用每个主要因素中的子要素来进行计算。例如，某些测试管理工具（在 Quality Center 的新版本里）

---

6 参见 Rick Craig 的“Systematic Software Testing”一书。



就支持这种计算。在这些公式里，我们可以给一些要素加大权重从而使其在风险优先级总分里比其他要素占更多的分数。

除了通过计算风险优先级数字来决定测试顺序以外，我们也需要用风险要素来安排测试工作。我们可以在许多方法中使用这些要素来获得测试范围。我们可以尝试使用另一种公式。例如，我可以用风险优先级数字乘以它给定的设计、实现和测试执行的时间。或者我们可以用定性的方法来匹配测试范围和风险优先级数字，这样可以根据测试人员的判断做出一些调整。

如果您确实选择使用公式来计算，需要注意根据历史数据进行调整。或者，如果在您的测试中是时间固定的，可以基于风险优先级数字来进行公式计算，根据风险成比例的分配测试工作。

相比公式而言，一些人更喜欢使用表格来从各要素获得综合风险优先级。表 3-2 给出了一个实例。

表 3-2 风险优先级表格

		影 响				
		很高	高	中	低	很低
可能性	很高	很高	很高	高	高	中
	高	很高	高	高	中	中
	中	高	高	中	低	低
	低	高	中	低	低	很低
	很低	中	中	低	很低	很低

首先和往常一样评估风险的可能性和影响，然后用表格根据风险项的可能性和影响评分得出其综合风险优先级。注意这个表格和我们前面提到的两种构建方法不同。现在通过不同风险优先级数字和风险优先级评分（从很高到很低）的匹配可以看出加法或乘法的计算方法是否与这个表格更一致。

和刚才讨论过的公式一样，您需要根据历史数据对表格进行调整。同时也应当根据利益相关者对每个风险的判断灵活地运用这种方法，允许综合风险优先级存在一定的偏差。

从表 3-3 可以看到我们不仅可以通过表格得出综合风险等级，还可以做一些类似测试范围这样的处理。根据风险优先级等级，可以用类似表 3-3 这样的表格来分配测试工作。你可能想花一点时间来学习一下这个表格。

表 3-3 测试范围表格

风险优先级	测试范围	注 释
很低	无	仅仅当在其他测试中发现该风险项的缺陷时进行缺陷报告
低	机会性的测试	平衡其他测试，仅仅在有机会的时候用最小的工作量来探索风险项
中	粗略的测试	运行少量的测试，针对风险项抽样测试最感兴趣的条件
高	广泛的测试	运行中等数量的测试，针对风险项测试覆盖许多不同的条件
很高	详尽的测试	运行大量测试，测试覆盖许多条件组合和条件变化



### 3.2.17 利益相关者参与

在本章的一些场合我已经提到过利益相关者参与风险管理的重要性。在上一节里，我们提到了各种风险识别和风险分析的技术。而让合适的利益相关者参与也很重要，甚至比技术选择更加重要。完美的技术缺少足够的利益相关者参与通常很难有所作为，但是一个不完美的技术如果有所有的利益相关方的支持和参与，它经常可以生成有用的信息并指导测试。

最重要的是我们拥有一个代表所有利益相关者的跨职能的团队，这些利益相关者对测试和质量感兴趣。这意味着我们至少有两个常用的利益相关者团队。其中一个利益相关者团队由那些理解客户和/或用户需求和兴趣的人组成，另一个利益相关者团队由对系统技术细节有深刻理解的人组成。我们也可以让商务利益相关者、项目投资者等一起参与风险管理。通过合适的参与者组合，良好的基于风险的测试过程可以帮助我们收集信息，并且让大家在哪些不需要测试、哪些需要测试、如何安排测试顺序和如何安排测试工作这些问题上达成一致。

我不能夸大利益相关者参与的价值。但如果缺少利益相关者的参与，至少会对风险识别和风险分析带来两项重大的功能障碍。首先，很难对优先级或工作安排达成一致。这意味着人们将在事实发生之后对你的测试进行事后评价。其次，你会发现在测试执行阶段甚至交付之后发现已识别的风险存在许多不足或者在风险级别评估中存在错误，这是因为过程的参与者有限造成的。

我们应当始终尝试让所有利益相关者参与风险管理。经常遇到的情况是并非所有利益相关者都能够或者愿意参与风险管理，在这种情况下，一些利益相关者可以作为其他利益相关者的代理。例如，在面向大众市场的软件研发中，市场团队可能要求小部分潜在的客户帮助识别那些可能严重影响他们使用的潜在缺陷。在这种情况下，这部分潜在客户就作为整个最终客户群的代表。另一个例子，IT 项目中的风险分析会话可能很痛苦，因为我们会对关于哪些可能出错以及会变得多糟糕这些话题进行讨论，有时我们可以让商务分析师代表用户参与这个过程，从而使用户免于参与这些可能令人痛苦的风险分析会话。

### 3.2.18 基于风险的测试与失效模式和影响分析练习 2

HELLOCARMS 是一个假设项目，它取自于 RBCS 帮助测试的一个真实项目，请阅读这个项目的系统需求文档。

如果在教室里学习，请将大家分成 3~5 个小组。如果您是单独学习，需要独自完成这个练习。请对这个项目进行质量风险分析。为了帮助管理这个练习所需的时间，这里只需要对功能质量特性进行风险识别和评估，请使用下面的模板。

为了合理分配练习时间，建议用 30 分钟识别质量风险，然后用 15 分钟评估每个风险的风险级别。

如果在教室里学习，当每一个小组完成了他们的分析后，对分析结果进行讨论，如表 3-4 所示。

表 3-4 功能质量风险分析模板

编号	质量风险	可能性	影响	风险优先级数字	测试范围	需求追溯
1.1.000	功能：适用性					
1.1.001	[在这部分列出适用性相关的功能风险]					
1.2.000	功能：准确性					
1.2.001	[在这部分列出准确性相关的功能风险]					
1.3.000	功能：互操作性					
1.3.001	[在这部分列出互操作性相关的功能风险]					
1.4.000	功能：安全性					
1.4.001	[在这部分列出安全性相关的功能风险]					
1.5.000	功能：合规性					
1.5.001	[在这部分列出合规性相关的功能风险]					

### 3.2.19 基于风险的测试与失效模式和影响分析练习 2 参考答案

表 3-5 是我对练习 2 的解答，随后是两项副产品的列表。一项是在分析过程里发现的项目风险列表，另一项是在分析过程里发现的需求文档缺陷。

作为质量风险分析的第一步，我核对了每一项功能需求并且对其识别了一个或多个风险项。我假设需求优先级可以很好地代表风险的影响，因此可以使用需求优先级代表风险的影响进行风险分析。即使用了所有这些简便方法，我仍然花了 1 个小时才完成这个练习。

表 3-5 HELLOCARMS 功能质量风险分析

编号	质量风险	可能性	影响	风险优先级数字	测试内容	需求追溯
1.1.000	功能：适用性					
1.1.001	拒绝所有房屋净值贷款申请	5	1	5	详尽的测试	010-010-010
1.1.002	拒绝所有房屋净值信用贷款申请	5	1	5	详尽的测试	010-010-010
1.1.003	拒绝所有反向住房按揭贷款申请	5	1	5	详尽的测试	010-010-010
1.1.004	无法正确处理部分房屋净值贷款申请	3	1	3	详尽的测试	010-010-190
1.1.005	无法正确处理部分房屋净值信用贷款申请	3	2	6	广泛的测试	010-010-200
1.1.006	无法处理部分反向住房按揭贷款申请	3	3	9	广泛的测试	010-010-210



续表

编 号	质 量 风 险	可能性	影响	风险优先 级数字	测试内容	需求追溯
1.1.007	无法处理部分组合产品(例如房屋净值和信用卡)	3	4	12	粗略的测试	010-010-220
1.1.008	无法处理部分住房按揭贷款申请	3	5	15	粗略的测试	010-010-230
1.1.009	无法处理部分预先批准的申请	3	4	12	粗略的测试	010-010-240
1.1.010	脚本对部分字段和屏幕不可用	4	2	8	广泛的测试	010-010-020
1.1.011	客户源数据未被收集	3	2	6	广泛的测试	010-010-030
1.1.012	没有很好地定义客户源数据分类	2	2	4	详尽的测试	010-010-030
1.1.013	在输入字段中接受无效数据	1	1	1	详尽的测试	010-010-040
1.1.014	现存交易过程没有被正确地显示和/或没有被正确地处理	3	1	3	详尽的测试	010-010-050 010-010-100
1.1.015	交易过程中的放款明细没有被传到贷款文件打印系统	2	1	2	详尽的测试	010-010-050 010-010-100
1.1.016	需要偿还的贷款被包含在债务收入比计算中	4	3	12	粗略的测试	010-010-110
1.1.017	无法继续未完成的/被中断的申请	3	2	6	广泛的测试	010-010-060
1.1.018	没有询问申请者现存关系	5	2	10	广泛的测试	010-010-070
1.1.019	现存的申请者关系没有被传到GLADS	3	2	6	广泛的测试	010-010-070
1.1.020	贷款状态信息在初始化后丢失	3	2	6	广泛的测试	010-010-080
1.1.021	无法干净地退出或关闭应用(例如, 必须关闭浏览器)	4	3	12	粗略的测试	010-010-090
1.1.022	无法通过客户 ID 查到其已有的申请	4	4	16	机会性的测试	010-010-120
1.1.023	对于超过\$500 000 的贷款并没有移交审批	4	1	4	详尽的测试	010-010-130
1.1.024	自动拒绝超过\$500 000 的贷款	3	1	3	详尽的测试	010-010-130
1.1.025	对于财产估价超过\$1 000 000 的并没有移交审批	4	2	8	广泛的测试	010-010-140
1.1.026	对于财产估价超过\$1 000 000 的自动拒绝申请	3	2	6	详尽的测试	010-010-140
1.1.027	电话营销输入性操作在应支持的范围内无法正常工作	1	2	2	详尽的测试	010-010-150
1.1.028	电话营销输出性操作在应支持的范围内无法正常工作	1	2	2	详尽的测试	010-010-150
1.1.029	不支持代理商和其他商务伙伴的品牌化	4	2	8	广泛的测试	010-010-160
1.1.030	未接受培训的用户(例如最终客户)无法通过因特网进入系统	4	3	12	粗略的测试	010-010-170

编 号	质 量 风 险	可能性	影响	风险优先 级数字	测试内容	需求追溯
1.1.031	不支持零售银行业务的产品运营	4	4	16	机会性的 测试	010-010-180
1.1.032	不支持灵活的价格实施方案	4	5	20	机会性的 测试	010-010-250
1.2.000	功能：准确性					
1.2.001	错误地将产品提供给不具备资格的客户	3	1	3	详尽的测试	010-020-010
1.2.002	没有将产品提供给具备资格的客户	4	1	4	详尽的测试	010-020-010
1.2.003	申请确定和 Globobank 的信用方针不一致	3	1	3	详尽的测试	010-020-010
1.2.004	基于风险的定价方式出现了对信用评分、贷款质量以及收入负债比的错误计算	4	1	4	详尽的测试	010-020-020
1.2.005	新的放款没有记入信用评分	3	2	6	广泛的测试	010-020-030
1.2.006	定价加成计算不正确	3	3	9	广泛的测试	010-020-040
1.2.007	没有对政府退休金收入进行适当处理	3	1	3	详尽的测试	010-020-050
1.2.008	没有捕捉到额外收入的持续时间	4	3	12	粗略的测试	010-020-060
1.3.000	功能：互操作性					
1.3.001	无法将信息从 GloboRainBQW 导入 HELLOCARMS	2	2	4	详尽的测试	010-030-010
1.3.002	HELLOCARMS 和评分系统拒绝联合申请	3	1	3	详尽的测试	010-030-020
1.3.003	HELLOCARMS 和评分系统无法处理/解决在联合申请中的重复信息	1	2	2	详尽的测试	010-030-030
1.3.004	HELLOCARMS 交易过程中与贷款文件打印系统交互失败	3	1	3	详尽的测试	010-030-040 010-030-070
1.3.005	贷款状态信息在贷款文件打印系统到 HELLOCARMS 的传递过程中丢失或出错	5	2	10	广泛的测试	010-030-050 010-030-140
1.3.006	如果评分系统显示存在未清偿的破产或赎回权丧失，HELLOCARMS 无法继续处理申请	4	3	12	粗略的测试	010-030-060
1.3.007	HELLOCARMS 到贷款文件打印系统的政府退休金收入信息传输失败	5	1	5	详尽的测试	010-030-080



续表

编 号	质 量 风 险	可能性	影响	风险优先 级数字	测试内容	需求追溯
1.3.008	HELLOCARMS 的申请信息没有被正确地传送到评分系统	3	1	3	详尽的测试	010-030-090
1.3.009	HELLOCARMS 无法正确地收到来自于评分系统的信息	3	1	3	详尽的测试	010-030-100
1.3.010	确认请求没有按评分系统要求进行排序	4	2	8	广泛的测试	010-030-110
1.3.011	临时批准的客户已接受的贷款没有传递到贷款文件打印系统	5	2	10	广泛的测试	010-030-120
1.3.012	被拒绝的申请信息没有被传到贷款文件打印系统	5	2	10	广泛的测试	010-030-130
1.3.013	在贷款文件打印系统中的贷款信息修改不能返回到 HELLOCARMS	5	2	10	广泛的测试	010-030-140
1.3.014	不支持计算机和电话系统集成	3	5	15	粗略的测试	010-030-150
1.3.015	申请者现存关系信息没有被传到 GLADS	4	3	12	广泛的测试	010-010-070
1.4.000	功能：安全性					
1.4.001	缺少达成一致的安全性需求	1	2	2	详尽的测试	010-040-010
1.4.002	“创建”和“最终修改”的审计轨迹信息丢失	3	1	3	详尽的测试	010-040-020
1.4.003	外包的电话营销的工作人员可以看到真实的信用评分和其他应保密的信息	3	2	6	广泛的测试	010-040-030
1.4.004	因特网应用无法保证在遭到恶意攻击时仍能工作	1	2	2	详尽的测试	010-040-040
1.4.005	因特网应用无法保证在遭到无意攻击时仍能工作	3	2	6	广泛的测试	010-040-040
1.4.006	不允许匿名浏览因特网	2	4	8	广泛的测试	010-040-050
1.4.007	反欺诈检查太宽松	3	1	3	详尽的测试	010-040-060
1.4.008	反欺诈检查太严格	3	1	3	详尽的测试	010-040-060
1.5.000	功能：合规性					
1.5.001	[在这部分列出合规性相关的功能风险]					

### 3.2.19.1 项目风险副作用

在准备质量风险分析文档的过程中，我发现了一个需求中的项目风险。那就是，考虑到缺少清楚的有关安全性需求文档（参见 010-040-010），支撑项目进度的必要的基础结构很可能无法完成。你注意到了多少项目风险呢？

### 3.2.19.2 需求缺陷副作用

在准备质量风险分析文档的过程中，我发现了以下需求缺陷：

1. 对于 010-010-150，支持的州、省和国家有哪些？
2. 对于 010-010-150，语句 all support States 应为 all supported States。
3. 对于 010-020-020，语句 dept-to-income 应为 debt-to-income。
4. 一些需求的编号重复，例如 010-030-140。
5. 有些需求没有被定义优先级，例如 010-030-150。
6. 对于 010-040-010，被提到的达成一致的安全性方法究竟是什么？
7. 银行无疑有许多法规的合规性需求，但该需求中没有直接提到任何法规的合规性需求。

你注意到了多少需求中的问题呢？

## 3.2.20 失效模式和影响分析（FMEA）

最后一个也是最正式的基于风险的测试技术是 FMEA。这种技术起先是作为质量设计技术进行开发的，但是你可以把它扩展应用于基于风险的软件和系统测试。和非正式技术一样，我们需要识别质量风险项，在这里被它称为失效模式。相比非正式方法而言，我们倾向于在这种方法里做得更加细致。这其中一部分原因是在识别失效模式以后，我们就会接着识别那些可能对用户、客户、社会、业务和其他项目利益相关者有影响的失效模式影响。

这种技术在精确度和严谨性方面有它的优势。当我们合理地使用这种技术时，相比较其他的技术而言，重要的质量风险被遗漏的可能性更低。危害分析技术具有类似的精确度，但是它不适用于复杂度高的项目，因为危害分析技术需要分析引发风险的上游的危害，而这是比较困难的。失效模式和影响分析（经常也被称为 FMEA 或 fuh-me-uh）对下游的影响进行分析相对容易，这也使得这种技术更为常用。

### ISTQB 术语

**失效模式和影响分析（FMEA）：**一个系统的进行风险识别和标识可能的失效模式的系统方法，用来预防失效的发生。

**失效模式，影响及危急程度分析（FMECA）：**FMEA 的扩展，除了基本的失效模式和影响分析，作为对 FMEA 的补充，还包括危急程度分析。危急程度分析技术用于分析失效模式发生的可能性及其后果的严重性并将其制成图表。对于具有高可能性和严重性的失效模式优先采取补救措施。

然而，这种高精确度和严谨性也有它的缺点。它会产生大量的输出，带来繁重的文档工作。这些文档工作不仅存在于最初的分析过程中，而且也存在于项目的分析维护阶段以及随后的项目里。另外这种方法也很难掌握，需要进行许多的实践。如果你一定要学习使用 FMEA，最好选择另一个项目，从运用非正式的质量风险分析技术开始。或者你也可以在这个项目上先进行非正式的质量风险分析，完成以后升级到 FMEA。



我已经在许多项目中使用过 FMEA。对于高风险或保守的项目，我绝对会考虑使用这种技术。但是对于混乱的、快速变化的或者原型项目我不会考虑使用 FMEA。

失效模式和影响分析起初是为了预防在设计和实施工作中的缺陷而开发的。我最早在 D.H. Stamatis 的“Failure Mode and Effect Analysis”一书里看到了这个技术并在 20 世纪 90 年代中期决定根据我的客户经验把这种技术应用到软件和硬件/软件系统里。随后，在 1999 年，我在我的第一本书“Managing the Testing Process”里讨论了这种技术。据我所知，这是第一次提出将 FMEA 技术应用于软件测试。我在“Critical Testing Processes”一书中进一步讨论了这种技术。所以，我不能说是我发明了这项技术，但是我可以说是首先在软件测试领域推广普及 FMEA 技术的人。

失效模式和影响分析存在许多不同的形式。其中一种就是失效模式、影响和危害程度分析（FMECA 或 fuh-me-kuh）。在 FMECA 里，每个影响的危害程度和其他影响风险级别的因素一起进行评估。

当 FMEA 被应用于软件时，还有另外两种变化形式，至少在命名上有所不同。这就是软件失效模式和影响分析以及软件失效模式、影响和危害程度分析。实际上，我经常听到人们在软件和硬件/软件系统里使用 FMEA 和 FMECA 技术。在本书中我们集中讨论 FMEA 技术，危害程度分析中的变化则是次要的，我们可以忽略这部分内容。

### 3.2.21 用失效模式和影响分析进行质量风险分析

FMEA 是一种迭代的方法。换句话说，在整个过程中都会反复对剩余风险进行重新评估。因为这项技术最初是作为设计和实施技术来开发的，所以在理想状况下应在项目早期就开始使用这项技术。

和其他形式的风险分析一样，我们希望测试分析师和测试管理人员参与帮助过程和生成 FMEA 文档。因为这个文档可能会很复杂，参与的测试人员对它们的目标和应用的理解就变得至关重要。和其他形式的风险分析一样，测试分析师、测试管理人员和其他参与者都应当运用他们的知识、技能、经验和独特的视野，根据 FMEA 方法进行风险分析。

就如我所提到的那样，FMEA 和它的变化形式并不适合所有的项目。但是在合适的项目中我们应该运用这种技术，因为它精确且彻底。遇到以下情况 FMEA 特别适用：

- 软件、系统或综合系统是关键的和风险失效必须减到最少。例如，航空软件、工业控制软件和核反应控制软件。
- 以强制降低风险或者以法规标准要求为准的系统，例如，医疗系统或那些以 ISO 61508 为准的软件。
- 不可以延期的项目，因为不可延期，所以管理层决定投入更多从而在项目早期尽可能多地消除缺陷。这不仅包括在测试中运用 FMEA，也包括在设计和实施中采用 FMEA。
- 复杂且安全关键的系统。这类系统需要周密分析从而定义特别的测试考虑、运行限制和设计决定。例如，一个战场指挥、通信和控制系统将和其他系统一起协同工作于千变万化的现代战争，这样的系统很适合使用 FMEA。

如我在前面提到的,在必要的时候你可以先使用非正式的质量风险分析技术,然后在此基础上扩展运用 FMEA,从而提高精确度等各要素。

因为 FMEA 技术起初并非为测试,而是为设计和实施而开发的,而且它是迭代的,所以你应当在过程的早期就计划 FMEA 活动。即使过程早期只具有初步的、高级别的信息时也需要如此。例如,只有市场需求文档或者甚至项目章程时,就可以开始进行 FMEA。当更多的信息和决定被明确后,您可以根据这些增加的细节进一步改进 FMEA。

另外,您可以在任意系统或软件分解级别上运用 FMEA。换句话说,您可以在一个系统上运用 FMEA,集成测试时在系统模块子集中运用 FMEA,或者甚至在一个单独的模块或组件上运用 FMEA。我已经分别在这些情况下运用过 FMEA。

无论从系统级别、集成级别还是组件级别开始,过程是相同的。首先,在每个功能、质量特性或者质量风险分类中识别失效模式。失效模式顾名思义就是一种可能失效的方式。例如,如果我们正在考虑一个电子商务系统的安全性,那么一个失效模式可以是“允许非法访问客户信用卡数据”。迄今为止,这可能听上去很像非正式质量风险分析,但在下一步里您会发现它们有所不同。

下一步我们会尝试识别每个失效模式可能的原因。在前面我们讨论过的非正式技术里并不包含这个步骤。为什么我们要这么做呢?记住,FMEA 起初是一个设计和实施的工具。通过尝试识别失效的原因,我们可以在设计外定义那些原因并且避免把这些问题引入到实施阶段。继续上面电子商务的例子,“允许非法访问客户信用卡数据”失效模式的一个原因可能是“信用卡数据没有被加密”。

下一步也是 FMEA 独有的,那就是对于每个失效模式,我们识别其可能造成的影响。这些影响可以是有关系统本身、用户、客户、其他项目和利益相关者,甚至是整个社会(记住这种技术经常被应用于像核反应控制系统这样的安全关键系统,一旦发生失效整个社会都有可能受到影响)。再次用电子商务这个例子,“允许非法访问客户信用卡数据”失效模式的影响可以是“盗刷客户信用卡”。

根据失效模式、原因和影响这 3 个要素,我们可以接下来评估风险级别。过一会儿我们将会提到这是如何工作的。我们也可以评估危害程度。在上面的电子商务例子中,我们可以说信用卡信息泄露的危害程度是严重的。

现在针对每种失效模式,我们可以决定采取哪种类型的风险缓解或风险降低步骤了。在非正式的质量风险分析方法里,我们把自己限制在定义测试范围里,而在 FMEA 里,假如我们让合适的人参与了过程,那么我们同时也可以指定设计和实施步骤。还是上面的电子商务为例,一种风险缓解步骤可以是“加密所有信用卡数据”,而一种测试步骤可以是“加密功能突破测试”。

注意这个例子强调了 FMEA 技术中的迭代要素。加密这个风险缓解步骤降低了失效模式的可能性,但是它引入了新的失效模式原因,例如“加密中使用了弱密钥”。

我们不仅在过程中进行迭代,而且有规律间隔地在生命周期中进行迭代。当我们获得新的信息和执行风险缓解步骤时,我们会进一步细化失效模式、原因、影响和风险缓解行为。



### 3.2.22 决定风险优先级数字

让我们回到风险要素和总体风险级别这个话题上来。人们在 FMEA 里一般用风险优先级数字来表示总体风险级别，或称为 RPN。

在运用 FMEA 时，通常使用 3 种风险要素来决定风险优先级数字：

- 严重性。指根据失效模式本身和影响来评估失效模式对系统的影响。
- 优先级。指根据效果来评估失效模式对用户、客户、业务、利益相关者、项目、产品和社会的影响。
- 发现可能性。指对系统中的问题在没有采取任何附加缓解措施的情况下未被发现的可能性进行评估。这需要考虑失效模式的原因和失效模式本身。

人们在运用 FMEA 时经常把这些风险因素用数字来表示。可以用 1~10 来度量，也可以用 1~5 来进行度量，另外也可以选择用升序或降序来进行度量。如果每种要素都使用同样的标尺来度量，那么可以用升序或降序来度量。换句话说，1 可以是指最高风险或者最低风险。如果你使用 1~10 的标尺来度量，那么降序意味着 10 是最低风险。如果你使用 1~5 的标尺来度量，那么降序意味着 5 是最低风险。相反的，如果你使用升序，那么 10 或 5 意味着最高风险。就我个人而言，我觉得不应该用任何粒度超过 5 分制的标尺来进行度量，除非我能够很准确地分辨出 9 和 10 或 2 和 3 之间的区别。这看上去就像我在就风险级别细节欺骗我自己和其他人一样。尝试达到如此高的精确程度也会增加利益相关者在风险分析过程中的争论时间，经常得不偿失。

就如我前面提到的那样，用这 3 种要素来决定总体或合计风险度量，或称为风险优先级数字（RPN）。最简单也是最普遍的做法是将这 3 个要素相乘。当然，也可以将 3 要素相加，或者用更复杂的计算方法，包括使用加权计算从而强调其中的一个或两个要素。

如同前面讨论过的非正式技术中的优先级编号一样，FMEA RPN 也决定了我们投入风险缓解的工作级别。但是注意 FMEA 的风险缓解不仅仅限于测试，实际上，可以进行多种级别的风险缓解，特别是当 RPN 很严重时。

谈到测试中的失效模式，我们可以使用 FMEA RPN 来对测试用例进行排序，每个测试用例都可以用与其相关的风险中优先级最高的作为其 RPN。然后我们可以按照风险优先级顺序来对测试用例进行排序。

### 3.2.23 FMEA 的收益、成本和挑战

运用 FMEA 的好处有哪些呢？除了高精确度和彻底性会减少风险误评估或遗漏以外，FMEA 也带来了其他的好处。由于它需要对因为软件失效或使用错误所引起的预期系统失效进行详细的分析，因此我们会对潜在的问题有全面的认识。

如果在系统级别使用 FMEA 技术，我们可以对跨系统的潜在问题有具体的认识。换句话说，如果考虑到系统风险，包括可靠性、安全性和性能风险，我们会对系统风险有更深刻的理解。当然，这也需要投入大量的时间来理解整个问题和它的重要性。

如前面所说，FMEA 相对于其他讨论过的质量风险分析技术而言的另一个好处就是它可以被用来帮助指导设计和实施。FMEA 可以提供正当的理由来决定不做某件事，否

定某个设计决定，不用某个特定的方法实施或者不用某个特定的技术进行实施。

和其他任何质量风险分析技术一样，FMEA 分析可以把测试集中在系统中明确和重要的部分。但是因为它比其他的技術更加精确，所以造成的效果也相对更为精确。由于可以更细致地分析风险，因此在测试设计中，也可以选择实施更细致的测试。

当然，FMEA 也带来了成本和挑战。您必须强迫自己考虑用例、场景和其他所有可能影响失效次序的因素。因为分析的粒度很细，所以很容易造成孤立地考虑每一种失效模式，而没有考虑到其他相关部分。相信你可以也应当克服这个挑战。

多次提到过 FMEA 表格和文档的数量很大，这意味着开发和维护这些文档将会是一项昂贵和费时的工.作。

如同起初构思的那样，FMEA 是按照每个功能进行工作的。当分析一个组件或复杂系统时，可能很难定义出相互独立的功能。可以不仅仅是通过功能，也可以通过质量特性或质量风险分类来进行分析，我自己曾经通过这种方法成功解决了上面的问题。

最后，当预测原因时，可能很难从中间影响里区分出真正的原因。例如，假设我们正在考虑一个电子商务系统中的失效模式“外国货币单位交易被拒绝”，我们可能会列出一个原因比如“信用卡验证无法处理外国货币”，但是实际情况可能仅仅是我们没有让信用卡供应商开通外国货币处理，这只是一个简单的实施问题而非软件问题。一般而言，在前面讨论过的那些质量风险分析中并不存在这个问题。

### 3.2.24 FMEA 案例学习

在图 3-4 里可以看到一个质量风险分析文档实例。这是一个真实的项目案例。这个文档和我们所用的方法都遵照失效模式和影响分析方法。

如你所见，我们从图左边每个明确的功能开始，然后识别失效模式和它们可能造成的影响。我们根据影响、严重性和优先级来决定危急程度。同时列出了可能的原因，从而帮助我们在需求、设计和实施过程中预防缺陷。

接下来我们看一下检测方法，在这个项目里的很多地方我们都用到这些方法。发现可能性越小，表示失效模式躲过检测的可能性越高。我们根据严重性、优先级和发现可能性来计算优先级数字。发现可能性越小越糟糕。严重性、优先级和发现可能性的范围都是 1~5，因此风险优先级数字的范围是 1~125。

因为这个表格是按照风险优先级数字排序的，所以图 3-4 只显示了排在最靠前的最高级别的风险项。对于这些风险项，我们希望采取额外的检测和其他被推荐的风险控制活动。你可以看到我们已经针对这一点分配了额外的活动但是还没有分配所有者。

在和风险项相关的测试活动中，我们期望测试用例的数量、测试数据的数量和测试覆盖程度都和风险成正比。注意我们可以允许任何包含风险项的测试规程继承风险项的风险级别，从而根据风险级别将测试规程的优先级文档化。

### 3.2.25 基于风险的测试与失效模式和影响分析练习 3

在前面的练习里，我们为 HELLOCARMS 项目做了非正式质量风险分析。现在，我们要用 FMEA 方法对非正式质量分析进行调整。



失效模式和影响分析（质量风险分析）格式——按风险优先级数字（RPN）排序											
原始 FMEA——按 RPN 排序											
系统功能或特性	潜在失效模式-质量风险	潜在的失效引发的影响	严重吗？	严重性	潜在的失效起因	优先级	检测方法	发现可能性	风险优先级数字	建议采取的行动	谁/什么时候？
碎片交换文档	分割碎片失败	安全缺口	Y	1	程序错误	1	测试：调试跟踪：代码评审	1	1	测试：调试跟踪：代码评审	
碎片交换文档	大量碎片	数据丢失	Y	1	程序错误	1	测试：调试跟踪：代码评审	1	1	测试：调试跟踪：代码评审	
压缩兼容性	数据损坏	数据丢失	Y	1	程序错误	1	测试	2	2	测试	
压缩兼容性	系统挂起	数据丢失	Y	1	程序错误	1	测试	2	2	测试	
压缩兼容性	切割碎片不当	数据丢失	Y	1	程序错误	1	测试	2	2	测试	
互联网文档识别	识别不当	数据丢失	Y	1	程序错误	1	测试：调试跟踪：代码评审	2	2	测试：规则验证	
网络兼容性	网络破碎	数据丢失	Y	1	程序错误	1	测试	2	2	测试选择/改进网络覆盖率	
删除文件名	FS 损坏	数据丢失	Y	1	程序错误	1	测试：调试跟踪：代码评审	2	2	测试：调试跟踪：代码评审	
删除文件名	删除失败	安全缺口	Y	1	程序错误	1	测试：调试跟踪：代码评审	2	2	测试：调试跟踪：代码评审	

图 3-4 失效模式和影响分析案例学习

为了做到这点，有 3 个主要步骤：

1. 对于每个风险项（我们称之为失效模式）添加一个或多个潜在的原因和影响。
2. 对于每个风险项，根据原因和影响来评估严重性、优先级和发现可能性，度量标尺范围是 1~5，其中 1 代表最高风险，5 代表最低风险。
3. 通过严重性、优先级和发现可能性这 3 要素计算风险优先级数字。

做完之后，将 FMEA 文档和非正式文档进行比较，思考在文档规模和细节上都有哪些增加。

你无需修改“测试范围”和“需求追溯”两列。前者的修改需要大量的讨论，而后者不需要修改。

如果在教室里学习，请将大家分成 3~5 个小组。如果您是单独学习，需要独自完成这个练习。请使用下面的模板。为了合理分配练习时间，我建议用 45 分钟进行转换。

如果在教室里学习，当每一个小组完成了他们的转换后，对分析结果进行讨论，如表 3-6 所示。

表 3-6 功能质量风险分析模板

编号	失效模式	潜在原因	潜在影响	严重性	优先级	发现可能性	风险优先级数字
1.1.000	功能：适用性						
1.1.001	[在这部分列出适用性相关的功能风险]						
1.2.000	功能：准确性						
1.2.001	[在这部分列出准确性相关的功能风险]						
1.3.000	功能：互操作性						
1.3.001	[在这部分列出互操作性相关的功能风险]						
1.4.000	功能：安全性						
1.4.001	[在这部分列出安全性相关的功能风险]						
1.5.000	功能：合规性						
1.5.001	[在这部分列出合规性相关的功能风险]						

### 3.2.26 基于风险的测试与失效模式和影响分析练习 3 参考答案

表 3-7 是我对练习 3 的解答。

表 3-7 HELLOCARMS 功能质量风险分析

编号	失效模式	潜在原因	潜在影响	严重性	优先级	发现可能性	风险优先级数字
1.1.000	功能：适用性						
1.1.001	拒绝所有房屋净值贷款申请	逻辑错误评分系统失效	主要的业务损失	2	1	5	10
1.1.002	拒绝所有房屋净值信用贷款申请	逻辑错误评分系统失效	重大的业务损失	2	1	5	10
1.1.003	拒绝所有反向住房按揭贷款申请	逻辑错误评分系统失效	部分业务损失	2	1	5	10
1.1.004	无法正确处理部分房屋净值贷款申请	逻辑错误评分系统失效	部分业务损失	3	1	3	9
1.1.005	无法正确处理部分房屋净值信用贷款申请	逻辑错误评分系统失效	较小的业务损失	3	2	3	18
1.1.006	无法处理部分反向住房按揭贷款申请	逻辑错误评分系统失效	极小的业务损失	3	3	3	27



续表

编号	失效模式	潜在原因	潜在影响	严重性	优先级	发现可能性	风险优先级数字
1.1.007	无法处理部分组合产品(例如房屋净值和信用卡)	逻辑错误评分系统失效	极小的业务损失	3	4	3	36
1.1.008	无法处理部分住房按揭申请	逻辑错误评分系统失效	极小的业务损失	3	5	3	45
1.1.009	无法处理部分预先批准的申请	逻辑错误评分系统失效	极小的业务损失	3	4	3	36
1.1.010	脚本对部分字段和屏幕不可用	用户接口错误	一些错误的贷款处理	3	2	4	24
1.1.011	客户源数据未被收集	用户接口错误	市场数据丢失	1	2	3	6
1.1.012	没有很好的定义客户源数据分类	用户培训错误 用户接口错误	市场数据部分丢失	1	2	2	4
1.1.013	在输入字段中接受无效数据	用户接口错误	错误地拒绝和批准贷款	3	1	1	3
1.1.014	现存交易过程没有被正确地显示和/或没有被正确地处理	用户接口错误 逻辑错误	错误地拒绝部分贷款	3	1	3	9
1.1.015	交易过程中的放款明细没有被传到贷款文件打印系统	逻辑错误 贷款文件打印系统接口失效	缺少合适的客户约束	2	1	2	4
1.1.016	需要偿还的贷款被包含在债务收入比计算中	逻辑错误 评分系统失效	错误地拒绝部分贷款	2	3	4	24
1.1.017	无法继续未完成的/被中断的申请	用户接口错误 逻辑错误	浪费代理人时间 较小的业务损失	2	2	3	12
1.1.018	没有询问申请者现存关系	用户接口错误	市场数据丢失	2	2	5	20
1.1.019	现存的申请者关系没有被传到 GLADS	GLADS 接口失效	市场数据丢失	1	2	3	6
1.1.020	贷款状态信息在初始化后丢失	贷款文件打印系统接口失效 贷款文件打印系统失效	无法按照客户要求提供状态信息	3	2	3	18
1.1.021	无法干净地退出或关闭应用(例如,必须关闭浏览器)	逻辑错误 用户接口错误	浪费代理人时间	4	3	4	36
1.1.022	无法通过客户 ID 查到其已有的申请	逻辑错误 用户接口错误	浪费代理人时间 给客户带来不便	4	4	4	48
1.1.023	对于超过\$500 000的贷款并没有移交审批	逻辑错误 用户接口错误	错误地发放贷款	2	1	4	8
1.1.024	自动拒绝超过\$500 000的贷款	逻辑错误 用户接口错误	错误地拒绝贷款	2	1	3	6

续表

编号	失效模式	潜在原因	潜在影响	严重性	优先级	发现可能性	风险优先级数字
1.1.025	对于财产估价超过\$1 000 000 的并没有移交审批	逻辑错误 用户接口错误	错误地发放贷款	2	2	4	16
1.1.026	对于财产估价超过\$1 000 000 的自动拒绝申请	逻辑错误 用户接口错误	错误地拒绝贷款	2	2	3	12
1.1.027	电话营销输入性操作在应支持的范围内无法正常工作	逻辑错误 用户接口错误	部分业务损失	2	2	1	4
1.1.028	电话营销输出性操作在应支持的范围内无法正常工作	逻辑错误 用户接口错误	部分业务损失	2	2	1	4
1.1.029	不支持代理商和其他商务伙伴的品牌化	用户接口错误	部分业务损失	4	2	4	24
1.1.030	未接受培训的用户(例如最终客户)无法通过因特网进入系统	用户接口可用性问题 安全性问题	部分业务损失	4	3	4	36
1.1.031	不支持零售银行业务的产品运营	逻辑错误 安全性问题 分支接口错误	部分业务损失	4	4	4	48
1.1.032	不支持灵活的价格实施方案	逻辑错误	部分业务损失	4	5	4	40
1.2.000	功能: 准确性						
1.2.001	错误地将产品提供给不具备资格的客户	逻辑错误 评分系统失效	错误地发放贷款	2	1	3	6
1.2.002	没有将产品提供给具备资格的客户	逻辑错误 评分系统失效	错误地拒绝贷款	2	1	4	8
1.2.003	申请确定和 Globobank 的信用方针不一致	逻辑错误 评分系统失效	错误地拒绝和批准贷款	2	1	3	6
1.2.004	基于风险的定价方式出现了对信用评分、贷款质量以及收入负债比的错误计算	逻辑错误 评分系统失效	逻辑错误 评分系统失效	2	1	4	8
1.2.005	新的放款没有记入信用评分	用户接口错误 逻辑错误	错误地发放部分贷款	2	2	3	12
1.2.006	定价加成计算不正确	逻辑错误	欺骗客户或 Globobank	2	3	3	18
1.2.007	没有对政府退休金收入进行适当处理	逻辑错误 评分系统失效	错误地拒绝和批准贷款	2	1	3	6
1.2.008	没有捕捉到额外收入的持续时间	逻辑错误 评分系统失效	错误地拒绝和批准贷款	4	3	4	24
1.3.000	功能: 互操作性						



续表

编号	失效模式	潜在原因	潜在影响	严重性	优先级	发现可能性	风险优先级数字
1.3.001	无法将信息从 GloboRainBQW 导入 HELLOCARMS	GloboRainBQW 接口失效	市场数据丢失	1	2	2	4
1.3.002	HELLOCARMS 和评分系统拒绝联合申请	逻辑错误	部分业务损失	4	1	3	6
1.3.003	HELLOCARMS 和评分系统无法处理/解决在联合申请中的重复信息	逻辑错误	部分业务损失	4	2	1	4
1.3.004	HELLOCARMS 交易过程中与贷款文件打印系统交互失败	贷款文件打印系统接口错误	缺少合适的客户约束	2	1	3	6
1.3.005	贷款状态信息在贷款文件打印系统到 HELLOCARMS 的传递过程中丢失或出错	贷款文件打印系统接口错误	无法按照客户要求提供状态信息	4	2	5	30
1.3.006	如果评分系统显示存在未清偿的破产或赎回权丧失, HELLOCARMS 无法继续处理申请	逻辑错误	错误地拒绝贷款	2	3	4	24
1.3.007	HELLOCARMS 到贷款文件打印系统的政府退休金收入信息传输失败	贷款文件打印系统接口失效	错误地拒绝贷款	2	1	5	10
1.3.008	HELLOCARMS 的申请信息没有被正确地传送到评分系统	Scoring 接口失效	错误地拒绝和批准贷款	1	1	3	3
1.3.009	HELLOCARMS 无法正确地收到来自于评分系统的信息	贷款文件打印系统接口失效	错误地拒绝和批准贷款	1	1	3	3
1.3.010	确认请求没有按评分系统要求进行排序	Decisioning 接口失效	错误地拒绝贷款	1	2	4	8
1.3.011	临时批准的客户已接受的贷款没有传递到贷款文件打印系统	贷款文件打印系统接口失效	错误地拒绝贷款	1	2	5	10
1.3.012	被拒绝的申请信息没有被传到贷款文件打印系统	贷款文件打印系统接口失效	不遵守法规 诉讼 声誉受损	1	2	5	10
1.3.013	在贷款文件打印系统中的贷款信息修改不能返回到 HELLOCARMS	贷款文件打印系统接口失效	给客户提供错误状态信息	4	2	5	30
1.3.014	不支持计算机和电话系统集成	呼叫中心电话接口失效	浪费代理人时间 给客户带来不便	2	5	3	30

续表

编号	失效模式	潜在原因	潜在影响	严重性	优先级	发现可能性	风险优先级数字
1.3.015	申请者现存关系信息没有被传到 GLADS	GLADS 接口失效	市场数据丢失	1	3	4	12
1.4.000	功能：安全性						
1.4.001	缺少达成一致的安全性需求	防火墙失效 加密问题	不遵守法规 诉讼	2	2	1	4
1.4.002	“创建”和“最终修改”的审计轨迹信息丢失	逻辑错误	不遵守法规 诉讼 声誉受损	2	1	3	6
1.4.003	外包的电话营销的工作人员可以看到真实的信用评分和其他应保密的信息	加密问题 用户接口错误 权限逻辑错误	不遵守法规 诉讼 声誉受损	2	2	3	12
1.4.004	因特网应用无法保证在遭到恶意攻击时仍能工作	加密问题 用户接口错误 权限逻辑错误	不遵守法规 诉讼 声誉受损	2	2	1	4
1.4.005	因特网应用无法保证在遭到无意攻击时仍能工作	加密问题 用户接口错误 权限逻辑错误	不遵守法规 诉讼 声誉受损	2	2	3	12
1.4.006	不允许匿名浏览因特网	用户接口错误 权限逻辑错误	部分业务损失	4	4	2	16
1.4.007	反欺诈检查太宽松	逻辑错误 评分系统接口错误	错误地发放贷款	2	1	3	6
1.4.008	反欺诈检查太严格	逻辑错误 评分系统接口错误	错误地拒绝贷款	2	1	3	6
1.5.000	功能：合规性						
1.5.001	[在这部分列出合规性相关的功能风险]						

一些需要注意的要点：

1. 在“发现可能性”评分中主要使用了非正式分析中“可能性”评分，因为这两者的含义在大部分情况下是相同的。

2. 在“优先级”评分中主要使用了非正式分析中“影响”评分，因为这两者的含义在大部分情况下是相同的。

3. 根据对系统的影响来评定“严重性”，等级范围参照缺陷跟踪系统：

- 数据丢失。
- 功能丢失且没有迂回的解决方案。
- 功能丢失但有迂回的解决方案。
- 功能部分丢失。
- 表面的问题。



4. 非正式风险分析包括大约 900 个单词，但在 FMEA 表格里有大约 1400 个单词，数量增长超过了 50%。实际上，非正式风险分析转换中很可能限制了文字量的增加。一般正式风险分析技术中的信息量要比非正式技术多出许多。

### 3.2.27 基于风险的测试和测试过程

我们已经讨论了质量风险分析技术。和其他任何技术一样，我们必须把选中的质量风险分析技术集成到较大的测试过程和较大的软件或系统开发过程中去。表 3-8 显示了一种可以用来组织质量风险识别、评估和管理的一般过程，它可以用于基于质量风险的测试。<sup>7</sup>

表 3-8 质量风险分析过程

步骤编号	步 骤
1	识别关键测试和质量利益相关者。获得利益相关者参加质量风险分析的承诺
2	和关键利益相关者一起研究质量风险分析技术和方法。如果有合适的技术和方法，可以提议使用。对技术和方法达成一致
3	从关键利益相关者那里收集有关质量风险、与这些风险相关的失效模式、这些失效对质量的影响以及风险优先级的想法。识别出推荐的措施来缓解每个风险
4	报告任何在分析阶段从其他项目文档中识别到的早期缺陷，例如不良或缺失的需求文档和设计问题等
5	根据使用的技术将质量风险文档化。将文档交于利益相关者批准。迭代步骤 3~5 是最终确定质量风险、质量风险优先级和质量风险推荐措施的必要步骤
6	将质量风险分析文档存入项目文档库或配置管理系统。对质量风险分析文档进行变更控制

让我们一步步具体地看一下这个过程。识别出参与的利益相关者，这是基于风险的测试取得成功的关键所在。您需要一个跨职能的团队来代表所有对测试和质量感兴趣的利益相关者。这个团队的代表性越强，那么您遗漏关键风险项和错误估计相关风险级别的可能性越小。利益相关者通常分为两类。第一类主要包括客户和用户或者理解用户和客户需求 and 兴趣所在的人，他们可以看到潜在的业务相关的问题并且评估其影响。第二类主要包括理解具体技术细节的人，他们可以看到哪些部分可能出错以及出错的概率有多高。然后，选择一种技术。在前面的章节里，您应当已经学到了一些如何选择技术的方法。

用所选的技术识别质量风险项。评估每项风险的风险级别。可以用各种方式来进行风险识别和评估，例如单次会议、头脑风暴或类似技术，以小组或一对一方式进行面谈等。尽可能对每个风险项的评分达成一致意见，如果无法达成一致，那么将其上报到合适的更高的管理层。随后选择合适的风险缓解技术。记住这不仅仅包括在一个或多个级别上进行测试，同时还包括需求、设计和代码评审，防御性的编码技术，用于保证安全和高质量代码的静态分析等。

<sup>7</sup> 我的书“Critical Testing Processes”首次描述了这种过程。

交付副产品。风险识别和分析经常会找到需求、设计、代码或者其他项目文档、模型和交付物中的问题。这些副产品可能是这些文档中真实存在的缺陷、项目风险或者实施假设和建议。应当将这些副产品交给合适的人进行处理。

评审、修改和最终确定质量风险文档。

现在这份文档是一项有价值的项目工作成果了。应当将其保存至项目库并且对其进行变更控制管理。理想情况下,质量风险文档应当仅仅在参与的利益相关者想法发生变化且同意进行修改的情况下才进行变更。

也就是说,这份文档是会发生变更的。应当对风险评估进行定期修改。例如,在重要的项目里程碑如需求、设计和实现阶段完成时以及在测试级别入口和出口评审中对质量风险文档进行评审和更新。同时,当获得大量的新信息时,也需要重新评审和更新质量风险文档,例如在测试级别的第一个测试周期完成时。应当计划增加新的风险项并且重新评估已有风险的风险级别。

在整个过程中都需要注意保持协同工作。除了过程中的信息采集特征以外,建立共识也是非常重要的一个特征。无论是专注于业务还是专注于技术的参与者,他们都能够也应当在风险优先级排序和风险缓解策略选择中扮演重要的角色。这样每个人都参与测试活动且对其具有责任。

### 3.2.28 整个生命周期中的基于风险的测试

在基础级大纲中曾经讨论过一个基本原则,那就是测试和质量保证应当尽早介入。这个原则强调预防性的测试。预防性测试是分析式的、基于风险的测试的一部分。在非正式质量风险分析技术中隐含了这部分内容,而在 FMEA 中明确提到了预防性测试。

预防性测试意味着在测试执行开始之前进行风险缓解。这需要一个测试级别开始前对测试件的早期准备、测试环境的预测试、产品早期版本的预测试、坚持更严格的测试入口准则从而保证需求和设计的可测性、参加包括回顾会议在内的项目早期活动评审,同时应当参加问题和变更管理,并且监督项目进度和质量情况。

在预防性的测试里,我们将质量风险控制活动融入整个生命周期中。测试经理应当寻找机会用各种技术来控制风险,例如:

- 合适的测试设计技术。
- 评审和审查。
- 测试设计评审。
- 各个测试级别合适的独立性。
- 用最有经验的人员执行测试任务。
- 确认测试(再测试)和回归测试的策略选择。

预防性测试策略认为可以也应当用多种活动来缓解质量风险,其中包括许多通常被我们认为不是“测试”的活动。例如,如果需求有问题,那么我们就应当着手评审并改进需求的质量,而不是用有问题的需求来进行设计、编码和测试。

动态测试并非对所有类型的质量风险都有效。例如,我们可以通过代码评审这种静



态测试方法轻松地找到与不良代码相关的维护性问题，但是动态测试将只能揭示不可维护的代码所造成的结果，与此同时这会带来过多的回归测试。

在某些情况下，对于给定的风险项可以估计出一般测试技术和特定测试技术的风险降低有效性。例如，基于用例的功能测试不太可能大幅降低性能风险或者可靠性风险。

因此，在测试有效性级别不高的部分使用动态测试来降低风险是不合适的。质量风险分析可以在项目的早期采用，从而在很多时候可以使利益相关者在进行诸如系统测试和系统集成测试这样的动态测试之前意识到质量风险缓解的机会。

一旦我们真正开始动态测试执行，我们用测试执行来缓解质量风险。当测试中发现缺陷时，测试人员通过测试使这些缺陷暴露出来从而提供了在发布前修改这些缺陷的机会，这样可以降低风险。当测试中没有发现缺陷时，测试可以通过保证在某些条件下系统运行正常来降低风险。

我在前面提到过在基于风险的策略中可以用风险级别来定义测试的优先级。这可以有许多种不同的方法，而其中极端的两种方法是深度优先和广度优先。在深度优先方法里，所有的高风险的测试将先于低风险的测试运行，测试将严格按照风险顺序执行。在广度优先方法里，我们对所有已识别的风险进行抽样测试，我们会用风险级别进行加权，但同时我们会保证所有风险至少被测试覆盖一次。

当运行测试时，我们应当根据剩余风险度量和报告测试结果。在某个范围的测试覆盖率越高，剩余的风险也就越低。在某个范围发现的缺陷越少，剩余的风险也就越低。当然在执行基于风险的测试时，如果我们仅仅根据风险分析来进行测试，可能会留下盲点。因此，我们需要使用在预定义测试规程之外的测试来探寻我们是否有遗漏。而这就是基于经验的测试技术展现其价值的时候，在卷一关于高级测试分析师中详细地讨论了这个技术。

如果在测试执行阶段需要减少测试时间或者测试的投入量，可以用风险来指导我们的工作。如果剩余风险是可接受的，可以缩短我们的测试。无论什么测试，运行过的测试应当比未运行的测试更为重要。如果我们确实缩短了测试，那么这将会把剩余风险转移到用户、客户、帮助台和技术支持人员或操作人员身上。

假如我们确实有时间来继续执行测试，在这种情况下我们可以根据当前测试中得到的信息来调整风险分析，并且为进一步的测试周期调整测试。首先，修改测试分析。然后，重新对已有的测试进行排序，也有可能需要增加新的测试。我们应当如何来决定是否需要调整风险分析呢？下面是一些主要的考虑因素：

- 全新的或者变化很大的产品风险。
- 在测试中发现的不稳定或者易有缺陷的地方。
- 与缺陷修复相关的风险，特别是回归风险。
- 发现未预料到的缺陷集群。
- 发现遗漏业务关键部分。

如果有时间进行新增加的测试周期，首先应当考虑修改您的质量风险分析。您应当在项目达到每个里程碑时都对质量风险分析进行更新。

### 3.2.29 基本测试过程中的基于风险的测试

现在我们将讨论如何将风险分析和基于风险的测试融入基本测试过程中去。让我们从测试的总体方向或目标开始，这与测试计划有关。

如前所述，风险管理应当贯穿整个软件生命周期。组织可以使用测试方针文档或者测试策略文档来描述测试中的产品风险和项目风险管理过程。文档中也可以描述风险管理是如何被集成到各个测试级别且对其产生影响的。在后面的测试文档章节里，我们将会谈到测试方针文档和测试策略文档。

测试计划是 ISTQB 基本测试过程中第一个活动，同时也是主要的测试管理活动。当使用基于风险的测试策略时，风险识别和风险分析应当作为测试计划的基础在项目开始时就进行。不仅仅在敏捷或者迭代模型中应当如此，对于其他任何的软件开发生命周期模型也应当这么做。

#### ISTQB 术语

**主测试计划：**通常针对多个测试级别的测试计划。

**测试计划：**描述预期测试活动的范围、方法、资源和进度的文档。它标识了测试项、需测试的特性、测试任务、任务负责人、测试人员的独立程度、测试环境、测试设计技术、测试的进入和退出准则和选择的合理性、需要紧急预案的风险，是测试策划过程的一份记录。[IEEE 829]

如果我们已经做了风险分析，那么接下来我们可以在测试计划中描述各种风险缓解活动。我们可以在主测试计划中针对项目中的风险安排缓解活动，或者在级别测试计划中针对特定级别的风险安排缓解活动。某些风险最好在特定的级别中进行缓解，而某些严重的风险则需要多个级别中进行关注。

如前所述，风险级别决定了测试投入大小和测试顺序。测试计划中应当描述测试投入的分配情况以及测试的顺序。除了描述我们希望通过测试来缓解的产品风险以外，测试计划中还应当包含如何应对和管理那些可能影响到已计划测试的项目风险。

在测试计划中的退出准则以及其他部分会涉及很多关于风险的问题。例如，何时风险缓解才算足够，什么样的剩余风险级别是项目利益相关者可以接受的，针对特定的项目风险我们可以采取哪些应对步骤等。测试计划中应当提到这些以及其他风险相关的问题。本章后面，会更详细地讨论这个话题。

既然刚才我们提到如何将风险管理融入测试计划过程，那么让我给出一些相关的小提示。这些小提示可以被专门应用于风险识别和分析任务中，同时它们也可以被应用于测试计划中。

在过程本身这部分我要再次强调前文中已经提到的一点内容，那就是在风险识别和风险分析中让跨职能团队参与是非常重要的。这不仅仅对识别和分析质量风险具有意义，同时也对识别测试相关的项目风险以及寻找处理这些风险的办法具有重要的意义。否则，在测试活动的后期，你会惊讶和沮丧地发现外部依赖条件并不能满足，或测试团队以外



的项目参与者不能或者不愿为测试活动完成他们应该完成的事情。

### ISTQB 术语

**基于风险的测试：**在项目初始阶段使用的一种测试方法，用来减低产品风险的级别并通知利益相关者产品风险的状态。这种方法包括了产品风险识别和产品风险在测试过程中的使用。

对于质量风险，我推荐一种方法，这种方法分两步。首先识别出质量风险项，然后评估风险级别。推荐这种方法主要有两个原因。首先，头脑风暴和分析的心理活动是不同的，参与者会发现在这两者之间切换是很麻烦的。其次，因为风险级别是相对的，所以除非我们能够看到整个风险项列表，否则对每个风险项单独设定合适的风险级别是很困难的。

对于测试相关的项目风险而言，情况也是类似的。首先，要识别出需要处理的风险。当识别出这些风险之后，我们可以考虑针对每项风险应当采取的行动。

基于风险的测试带来的一个主要的潜在问题是我们需要管理其中大量的信息。所以在可能的情况下应当尽量简单。在危害分析和 FMEA 里，因为详细的信息是这些技术所必需的，所以我们必须花费时间和精力来管理这些信息。但是在更为非正式的技术里，应当尽可能减少需要管理的信息量以及相关的文档。我们可以仅在风险项涉及不同风险级别时才将其细化。

例如，假设我们在测试一个电子商务网站。我们可能识别出“系统对客户输入响应缓慢”这样一项质量风险项。但是响应缓慢这个风险在浏览网页和结账时所产生的影响是不同的。我们可能会将这个风险分成浏览响应缓慢和结账响应缓慢两个风险项。

另一种管理潜在的大量信息的方法是根据既定的项目限定和需求，使用可能的最轻量级的非正式技术。如果安全关键或者危害性问题需要如 FMEA 这样正式技术的，我们当然可以采用正式技术。但是在不需要的场合过度使用正式技术将可能削弱基于风险策略的总体支持效果并且可能浪费有限的测试资源。

让我们快速地回顾一下前面提到的 3 点内容。首先，一些人仍然把基于风险的测试作为专注于最大程度寻找缺陷的一种缺陷搜寻手段。这种方法仅仅考虑那些很可能引发缺陷的风险的技术层面。然而并非所有测试都应当专注于寻找缺陷，有些测试应当主要用于建立信心。因此，一定要考虑风险的业务层面并且对那些可能造成严重影响的缺陷进行测试，无论其发生概率是高还是低。

其次，基于风险的测试应当是预防性测试方法的一部分。这意味着我们应当尽可能早地在项目中开始风险识别和风险分析。

最后，牢记风险项和风险级别将会随着项目的发展而不断变化。因此当考虑如何将基于风险的测试融入过程中去的时候，一定要在关键的项目里程碑时重新调整风险分析、测试和项目。

### 3.2.30 基于风险测试的挑战

采用了基于风险的测试后，需要克服许多挑战。其中许多在测试计划活动中最为尖锐，但它们可能也的确在整个测试过程中都持续存在。让我们看一下其中一些关键的挑战并且给出一些克服它们的办法。

第一项挑战发生在风险分析阶段。作为一个引领质量风险分析工作的测试经理，我经常发现很难对那些与一个或多个风险项有关联的风险项的风险级别达成一致意见。一些人说可能性高，而另一部分人会认为可能性低。对于风险的影响大家也会有截然不同的看法。这可能导致无法最终确定质量风险分析结果并且降低基于风险的测试对剩余的测试过程的正面影响，因为总体优先级以及相关的测试顺序和潜在的测试选择优先级都是未知数。

有许多方法可以解决这些不一致的问题。如果您或者争议方中的一方在组织中有一些政治影响力的话，可以尝试影响这个决定。就我个人而言，我并不喜欢或不想被认作喜欢采用这种方法。我只在极少的情况下会真正使用我的政治权力（通常是命令）来进行风险评估。如果我有一种优先级评估方法可以让所有利益相关者对于每项风险都达成一致，那么我不会介意那到底是哪种优先级评估方法。

我可能会尝试去调和某些妨碍达成一致的情况。有时，大家并非对风险级别无法达成一致，而是对相关的测试工作量以及测试顺序无法达成一致。您也许可以询问争议双方是否可以在其他风险级别上做出改变而让所有风险的总体级别更加符合双方要求从而得到一个双方都能够接受的结果。

在一些情况下，不一致的根本原因是不知道评估的理由。换言之，您可以通过培训来克服这个挑战。您可以让合适的利益相关者加入风险评估过程中。例如，开发人员、系统管理员和技术支持人员会对诸如某些类型的缺陷的发生可能性有更强的洞察力，而销售、市场营销和业务分析员则对诸如某些类型的缺陷的影响更有发言权。

当然最后可以采取的办法是把不一致的问题上报给一些资深项目利益相关者那里。既然这至少对有些风险是有可能发生的，我们应当从一开始就安排一些人担任这样的角色来解开死锁。这个角色在某些项目或者组织里需要极大的智慧和耐性，因此应当要寻找合适的人选。因为您想要做的是达成一致，所以记住这个角色也需要所有利益相关者的同意，例如，他们同意接受这个人的决定。<sup>8</sup>

下一个挑战也始于风险评估，在任何的风险级别重估时都可能再次发生。有时很容易对大部分风险的风险级别达成一致，因为大家都同意每一个风险都具有高优先级！如我的一个客户所说，问题在于“当每个风险都具有高优先级时，实际上就没有优先级了。”当然，这会损害到基于风险测试的价值，因为在有限的测试资源和测试时间面前，现在我们无法用相关的风险级别来进行平衡了。

---

<sup>8</sup> 对于非阿拉伯宗教背景的读者而言，根据基督教徒、犹太教徒和伊斯兰教信徒的宗教书中记载，所罗门是一位 3000 年前的以色列国王，他能够聪明机灵以做出判决并得出正确的判决结果而著称。约伯是一个成功的虔诚的商人，它在上帝和魔鬼的斗争中忍受了极大的折磨，而最后他终于恢复了健康和成就。



也可以用上报来解决这个问题,但在此之前您可以尝试用一些技术来解决这个问题。一方面,要认识到所有项目都是质量、进度、范围和预算这些要素不断变化的连续统一体,我们需要对这些要素进行平衡。我们或是提前用计划的方式来慎重地、有分析性地、仔细地来进行平衡,或是用匆忙的、反应式的和混乱的方式来平衡。因此,如果在风险评估阶段你察觉到结果有“一切都是高优先级”倾向的时候,应当要求大家考虑一个问题,如果对每个评估的风险项进行彻底地测试大家将会牺牲什么。

大家是否愿意在项目中花费\$10 000、\$100 000、\$1 000 000 或更多来保证对风险项进行彻底地测试呢?大家是否愿意延后发布一天、一星期、一个月或一年来保证对风险项进行彻底地测试呢?如果无法保证所有的相关风险项都能被彻底地测试,大家是否愿意削减一个完整的项目功能或项目特性呢?如果答案是“无论要进行多少测试,不会多一分钱,不允许有任何发布延后同时也不能削减任何功能”,那么大家可以猜到什么呢?这意味着这个风险项不是一个高优先级的质量风险。

如果这个问题反复发生,那么另一个处理这问题的技术是采用风险分析方法来明确测试的成本。例如,用前面提到的测试的成本可以解决这个问题。

另一项挑战发生在整个基于风险的测试过程中。这就是如何让人们风险做出理智的决定。不幸的是,人类通常都无法理智地面对风险,人们经常是非理性的。

让我说一件轶事,我有一个和我一样经常出差的咨询师朋友。和我不同但和许多其他人相同的是他对飞行很恐惧。他的这种恐惧主要来源于他在拉瓜迪亚机场乘坐双引擎喷气式飞机出差时的经历,当时飞机出了问题,按照飞行员的说法是一次“飞鸟撞击”。在那次特殊的事件里,一只海鸥径直飞入了一个喷射口从而使其在起飞时熄火。这导致了被称为“起飞失败”的问题,而这种说法是对飞行中最危险的事件之一的温和说法。

现在过了许多年,我的这个朋友买了一辆漂亮的黑色保时捷运动跑车。有一天当我造访他在纽约附近的家时,他带我一起乘坐了他的这辆跑车。我记得我们在长岛高速公路交通繁忙的时候以超过每小时 120 英里的速度行驶。所以我想,“真有趣,这就是那个害怕飞行的人。”

记住在交通事故中的死亡率大约要比飞行事故中的死亡率高 100 倍以上。事实上,在工业化国家中,许多在汽车事故中受伤的人都不会因此而害怕开车,这包括我自己还有我知道的那些在汽车事故中严重受伤的人。

可以在 RBCS 网站 [www.rbcus.com](http://www.rbcus.com) 上阅读 Erik Simmons 的“风险观念”文章来简明而准确地了解这种情况发生的原因。概括而言,Erik 列举了一些会增加或降低风险的感知级别的要素。

风险带来的恐惧感觉会提升风险的感知级别,而精确或不精确的对风险可控制的感觉会降低风险的感知级别。

强加的风险会有更高的风险感知级别,而那些自愿性风险的感知级别则较低。

直接结果会提高风险感知级别,而延时的结果会降低风险感知级别。

对结果的了解程度低会提高风险感知级别,而对结果的了解程度高则会降低风险感知级别。

慢性结果会降低风险感知级别,而急性的结果则会提高风险感知级别。

相当自然的是，严重的结果会提高风险感知级别，而非严重的结果会降低风险感知级别。

如果在考虑中的风险和一些其他负面事件或其他事件有关联、有类似或者具有历史关系，那么这会提高风险感知级别，特别是如果相关的事件曾经有严重的后果而这个后果比在考虑中的风险的最可能的后果还要严重时。换言之，由于人类经常通过现有的、已经理解的概念类推来理解新的概念，然而，如果这种类推联系到负面事件，这会使人直接希望避免该风险。如果在讨论中的风险过去曾经确实成为过负面事件，特别是如果造成的后果十分严重时，这会提高风险感知级别。

最后，对先前或当前项目中的风险的熟悉程度和经验有助于降低风险感知级别。

注意我已经反复使用风险感知级别这个词语。这和真正的风险级别不同，而这就是问题所在。这是不理性的部分，但同时这也帮助我们更好地应对挑战。如果可以将这些非理性的部分公开化，让它们成为讨论的一部分，那么有机会让大家对风险变得更为理性。<sup>9</sup>

此外，虽然我把这些归为非理性的行为，但是注意从生物学的角度看这些是聪明的和适应的本能。想像一下 100 000 年以前，您自己站在非洲的平原上，看到一头大型斑点毛皮的像猫一样的动物从面前跑过。您从没有见过这样的动物，但是曾经看到过一头类似大小的外表像猫的动物，而上一次您看到它的时候它正在吃您的兄弟。那时，这头动物到底是一头非常可能要吃掉你的老虎还是不太可能来吃你的非洲猎豹已经不重要了，从这头动物身边跑开是聪明的做法。

另一项挑战存在于整个生命周期，那就是时间投入的问题。基于风险的测试需要预先的和持续的时间投入。所以确保你（作为测试经理）和其他的项目管理团队有明确的进行风险识别、风险评估和定期风险重新分析的时间和任务。

最后，有些风险分析技术面临适应变更的挑战。如果您在一个快速变化的项目中，例如采用非结构化的生命周期或者敏捷的方法时，您很可能需要更多地使用非正式技术，而不是正式技术。

### 3.2.31 基于风险的测试与失效模式和影响分析练习 4

运用 HELLOCARMS 的质量风险分析，设计一套功能测试用来进行 HELLOCARMS 功能集成测试和系统测试。系统集成测试级别应当寻找 HELLOCARMS 应用程序和其他数据中心应用程序一起工作时的缺陷并以帮助建立信心作为一个总体的目标。系统测试级别应当把寻找 HELLOCARMS 应用程序提供的必要功能中的缺陷并以帮助建立信心作为一个总体的目标。

最后，叙述测试执行中每个测试套件里对测试用例进行排序的总体指导方针。

如果在教室里学习，当每一个小组完成了他们的分析后，对分析结果进行讨论。

---

9 我在“Critical Testing Processes”一书中讨论了合理性和风险的这个问题。具体有关这个话题以及更广的关于风险的内容，我也推荐参阅“The Logic of Failure”、“Against the Gods”、“Freakonomics”和“True Odds”。



### 3.2.32 基于风险的测试和失效模式和影响分析练习 4 参考答案

根据执行过的质量风险分析，我创建了以下系统集成测试套件和系统测试套件列表。记住我们这里仅考虑功能测试，因为本题的质量风险分析仅专注于功能测试。

系统集成测试套件：

- HELLOCARMS/评分系统接口。
- HELLOCARMS/ LoDoPS 接口。
- HELLOCARMS/ GLADS 接口。
- HELLOCARMS/ GloboRainBQW 接口。

系统测试套件：

- 基本功能。
- 不良申请人。
- 端到端处理。
- 出错处理和恢复。
- 安全性。
- 一致性。

测试用例排序指导方针。

因为这个应用程序展现出功能范围大以及必须处理的不同的情况多这样的特点，我决定用广度优先的方法作为测试排序的指导方针。在广度优先的方法里，我们从所有已识别的风险中选择一个样本，用风险级别来对该样本进行加权处理，同时保证每个风险至少被覆盖到一次。

所以在测试执行开始之前，我们用来自于质量风险分析的可追溯信息来使得测试规程继承其追溯到的高风险质量风险项的风险优先级数字。然后我们可以将整个测试用例集合分成三部分：

- 高优先级：包括占总体计划测试工作 1/3 的测试规程子集。50%的测试规程的风险优先级数字是 1~5。35%的测试规程的风险优先级数字是 6~10。15%的测试规程的风险优先级数字是 11~15。
- 中优先级：包括占总体计划测试工作 1/3 的测试规程子集。35%的测试规程的风险优先级数字是 1~5。50%的测试规程的风险优先级数字是 6~10。15%的测试规程的风险优先级数字是 11~15。
- 低优先级：包括占总体计划测试工作 1/3 的测试规程子集。15%的测试规程的风险优先级数字是 1~5。15%的测试规程的风险优先级数字是 6~10。70%的测试规程的风险优先级数字是 11~15。

从而我们有了一个高优先级的测试周期，一个中优先级的测试周期以及一个低优先级的测试周期。在每个测试执行周期里，我们严格按照风险优先级顺序运行测试子集中的测试用例。

这种排序方法会根据测试发布时间表和测试集合的总体大小变化。例如，如果总体测试被安排在最后 6 个星期而测试发布被安排在每个星期接收一次，那么我们有 3 个两

周的周期,每个周期包括两次测试发布的接受。如果总体测试被安排在最后8个星期,那么我们就有4个子集而不是3个,这会显得更加合理。

你会注意到这种排序方法假设了测试规程将会根据风险优先级数字1~5、6~10和11~15大约地均分到3个子集里。如果这种假设不成立,那么计划应当被调整。但是,不管实际的测试规程根据风险优先级数字的组合如何,计划的基本目标是可达成的。

上面主要讲的是系统的测试工作,还有一些时间用于非系统的测试,这可以分为两类:

- 确认测试:测试我们在计划或非计划的测试发布里收到的任何进入测试环境的缺陷修复。
- 应对式测试:运用探索性测试,Whittaker攻击、缺陷分类、Hendrickson缺陷捕捉技术和其他非系统性的测试技术。(基础大纲中提到过这些技术,在本课程后续内容中会更详细地进行讨论。)一些反应式的测试将可以处理等级为“机会性的”或更低的风险以及没有在风险分析中被识别到的风险,这可以作为系统检查和质量风险分析的平衡。

每个测试周期中60%~70%的时间花在系统的测试上,而30%~40%的时间花在确认测试和反应式测试上。

### 3.2.33 FMEA 案例学习二

图3-5中展示了一个我和我的同事在一个项目里为客户准备的失效模式和影响分析的表格实例。这是从大约250行的工作表里取出的很小的一部分。你可以看到在最左边的一列里,我们从识别正在考虑中的特定的功能开始。如我前面提到的那样,失效模式和影响分析方法的一个问题是可能遗漏与特定功能无关的主要的风险分类。我们通过把应对突发的和系统的风险这部分包括在内来处理这个问题,例如性能、可靠性等。

在下一列里,我们识别特定的质量风险项。在第三列里,我们识别风险成为实际的事件后的影响(或者说造成的后果)。

您会注意到关键性这一列,这使得它成为FMEA的变种FMECA。再下一列是严重性,这里仍然指的是对系统造成的不良影响。和本例中其他的风险因素一样,严重性的等级为1~5,其中1代表最严重。

其后一列是潜在的失效模式的原因。在这个特定的风险分析里,我们并没有许多技术方面的参与。这限制了我们在这部分的见解。如果必须把这个特定的项目做完,我会尝试让更合适的利益相关者来一起参与。

然后我们可以看到优先级,这是对用户、客户、业务和其他利益相关者的影响。同样的,这部分等级范围也是1~5。

下一列是发现方法。同样,作为专注于测试的和测试人员准备的风险分析,我们对各种可能发现缺陷的测试级别进行识别。

再下一列是发现可能性,这是指现有的缺陷在没有明确的针对性测试的前提下不被觉察的可能性。同样的,这部分等级范围也是1~5。



CSA FMEA 表格										
系统功能或特征	潜在的失效模式 质量风险	潜在的失效影响	关键性?	严重性	潜在的失效原因	优先级	发现方法	发现可能性	风险优先级数	推荐行动
应用程序进入/退出										
	CSA 应用程序无法启动	数据/服务损失	Y	1	CSA 应用程序缺陷	1	CSA 字符测试	5	5	字符测试
	CSA 应用程序无法退出	数据/服务损失	Y	1	CSA 应用程序缺陷	1	CSA 字符测试	5	5	字符测试
	应用程序终止后客户端的应用程序/处理仍在运行	潜在数据/服务损失	Y	1	CSA 应用程序缺陷	1	系统测试	2	2	系统测试
	应用程序终止后服务器端的应用程序/处理仍在运行	潜在数据/服务损失	Y	1	CSA 应用程序缺陷	1	系统测试	2	2	系统测试
	客户端应用程序退出/停止机制失效	客户不满, 业务损失, 数据损坏	Y	1	CSA 应用程序缺陷	1	字符测试	3	3	字符测试
	置零后 CSA 弹出式通话记录失效	数据/服务损失	Y	2	CSA 应用程序缺陷	2	集成测试	3	12	字符测试
	强迫置零后 CSA 弹出式通话记录失效	数据/服务损失	Y	2	CSA 应用程序缺陷	2	集成测试	3	12	字符测试
	直拨时 CSA 弹出式通话记录失效	数据/服务损失	Y	2	CSA 应用程序缺陷	2	集成测试	3	12	字符测试
	CSA 无法显示从自动语音系统得到的通话原因	数据/服务损失	Y	3	CSA 应用程序缺陷	3	字符测试	3	27	字符测试

图 3-5 失效模式和影响分析案例学习二

现在我们来到了风险优先级数字,或者说 RPN。RPN 是通过计算获得的。在本例中,RPN 来自于严重性、优先级和发现可能性三者的乘积。

最后 3 列的内容是推荐的行动,行动负责人以及阶段安排,还有参考的和这个风险项相关的项目工作产品内容。

### 3.2.34 基于风险的测试和测试控制

接下去让我们顺着过程继续向前来到测试控制。一旦测试计划结束以后,风险管理应当继续并贯穿整个过程。这包括了风险重新识别和分析,同时还包括了采用各种措施来降低风险。我们可能在测试设计、测试实现或测试执行期间识别到新的风险。从其他项目活动中获得的新的信息可以帮助我们重新评估现存的风险级别。我们应当在测试执行阶段评估测试有效性和其他风险缓解活动。

这些活动中有一部分发生在项目里程碑时。如果我们根据需求规格说明识别和评估风险,当设计规格说明完成时我们应当对风险重新进行评估。如果我们在测试执行阶段项目的某个组件上发现大量的未预料到的缺陷,我们可以认为缺陷发生的可能性会高于我们评估时的想法。然后我们可以调高可能性和总体风险级别,增加针对这个组件所需运行的测试数量。

在本章的前面部分我提到过,我们可以根据风险对测试进行深度优先或者广度优先排序。但是很重要的一点是我们需要理解无论在测试执行中采用哪种方法,我们可能用尽分配的测试执行时间但还是无法完成所有的测试。而这是基于风险的测试让我们根据风险进行测试选择的原因。

我们可以根据剩余风险级别向管理层报告测试结果。管理层则可以决定延长测试或者将剩余风险转移到下游的用户、客户、帮助台和技术支持以及操作人员身上。稍后我们将再看一个如何根据剩余风险进行报告的例子。

我们很少会发现在测试完成以后还有剩余时间来进行额外的测试(但有时候这确实会发生)。当然,管理层可以决定就在那个时候提早结束项目。如果管理层决定要继续,那么我们可以根据风险重评估结果来增加新的测试周期。重评估可能会发现新的或变更的产品风险,系统中缺陷集中的地方(根据我们的测试结果)和缺陷修复中产生的新风险。我们可以通过对已经发现的问题的典型性分析把我们的测试调整到开发人员已经产生缺陷的部分。同时我们也可以针对那些缺少测试的部分增加一定的测试。

如果我们确实为了随后的测试周期而对风险进行了重评估并且完善了我们的测试,我们应当更新质量风险文档来体现为什么我们要这么做。同时可能也需要有相关利益者来进行批准。

除了必须在测试执行期间通过测试来管理产品风险以外,我们也必须控制测试相关的项目风险,因为这些风险可能影响到我们的测试执行。表 3-9 是一个实际的笔记本开发项目相关实例。

这个例子列出了 3 个可能影响我们测试运行的项目风险。第一个和笔记本上的光盘只读存储器潜在问题有关。如果发生这个问题,测试结束时间将会被迫推迟。你可以看



表 3-9 项目风险控制实例

风 险	影 响	引 发
在 C-Test 主板上的光盘只读存储器失效需要返工,从而造成测试延误	测试结束时间延迟到 11 月 28 日,为首个客户交付关键路径项	11 月 14 日
PC 卡初始化成功,但是无法在 Windows 系统下工作	如果这个问题是必须修复的,那么它必须及时被修复从而留下足够时间让测试人员能在 11 月 22 日前完成测试。假设测试需要 5 天,那么我们需要在 11 月 15 日前解决这个问题	11 月 15 日
“金牌候选” BIOS, Q3A11 无法在 11 月 19 日前进入测试	这里有两方面的风险:在 11 月 22 日前最后的三四天里测试可能被打断,“金牌候选” BIOS 可能没有经过足够的一致性测试就被交付进行测试。供应商没有在预先计划的 11 月 19 日交付新的 BIOS。这会增加上面已经提到的风险	11 月 19 日

到引发时间,这时我们将会知道这个风险是否会真的变成实际的事件的时间点。在某些情况下,如果到了引发时间这个风险真的成为实际的事件,那么这会“引发”(因此得名)一个应急计划。

下一个风险实际上是一个产品风险同时也是一个项目风险。注意这里的引发时间实际上是一个最后期限。

表 3-9 中最后一个风险和供应商无法按时交付 BIOS 有关(BIOS 是内建的帮助计算机启动的软件)。因为测试将会不足,所以这个项目风险已经造成了产品风险。

### 3.2.35 基于风险的测试结果评估和报告

通过在测试计划中识别和评估的风险以及测试设计和实现中达到的适当的测试覆盖度,我们可以根据风险减少量来追踪测试执行。换句话说,我们可以根据减少和未减少的风险来评估基于风险的测试结果。基于这个评估结果,我们可以根据减少和未减少的风险来报告基于风险的测试结果。

如前所述,在实际情况下这需要具备测试用例和缺陷到风险项的可追溯性。通过这种可追溯性,测试度量和报告可以包括以下部分:

- 风险:已经为其执行了一项或多项测试。
- 风险:已经有一项或多项相关测试失败。
- 风险:有一个或多个必须修复的相关缺陷。

根据这些结果,项目利益相关者可以对例如发布是否准备就绪这样的重要问题做出聪明的基于风险的结论。让我们看两个基于风险评估和报告的例子。

表 3-10 展现了一个表格式报告。在这个报告里,主要的风险区域被列在左边。一些在风险分析中被识别的主要风险区域被集中放在最后的“其他”区域中,可能因为这部分识别到的缺陷数量不多。这个报告根据缺陷发现数量进行排序。

中间的两列给出了缺陷发现数量和每个区域的相关百分比。右边的 3 列是各个风险区域计划的测试数量,执行的测试数量和计划测试的执行百分比。

表 3-10 例子：测试和风险

质量风险区域	缺陷数量	测 试			
		测试%	计划的测试数量	执行的测试数量	计划测试执行%
性能	304	27	3843	1512	39
安全性	234	21	1032	432	42
功能	224	20	4744	2043	43
可用性	160	14	498	318	64
接口	93	8	193	153	79
兼容性	71	6	1787	939	53
其他	21	2	0	0	0
总计	1107	100	12 857	5703	44

如你看到的那样，性能这部分的风险仍然相当高。在这部分我们已经发现了大量的缺陷，而且只执行了 1/3 多一点的测试。我们应当能够预见到在剩余的性能测试结束后可以发现更多的缺陷。接口这部分的风险（相对而言）并不高。我们已经运行了大部分的测试，而在这部分发现的问题相对要少得多。

对于某些人来说，无论用表格还是图来表示，表 3-10 中包含的细节还是太多了。当我告诉我的一个客户基于风险测试报告结果时，他说：“你看，这对我是有意义的，但是对于我的担任首席信息官的老板来说，如果结果不够简单的话他是不会注意看的。”

因此，图 3-6 是一个掌握了基于风险测试结果报告精髓的单一简单的图表。中间灰色部分代表所有相关测试被运行通过且没有发现必须修复缺陷的风险。浅灰色部分代表至少有一项相关测试没有通过且至少有一个必须修复缺陷的风险。暗灰色部分则代表剩下的那些没有已知的必须修复缺陷但测试还未完成的风险。

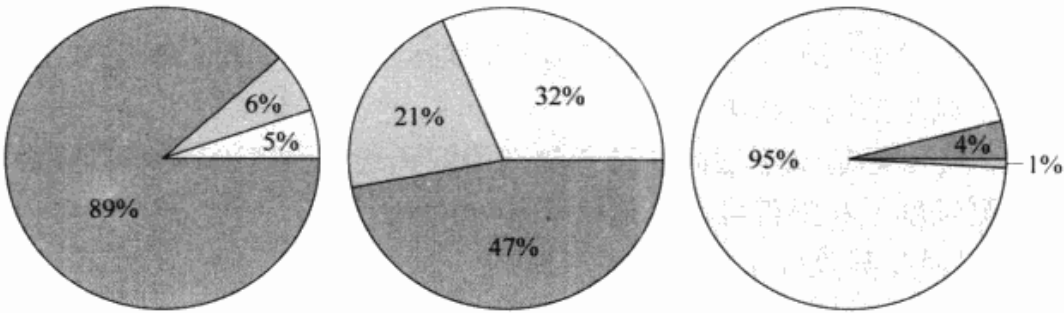


图 3-6 报告剩余风险级别

底下的时间轴是表示当我们开始测试时饼图完全是暗灰色的。随着测试执行的展开，我们可以看到浅灰色部分逐渐开始占据暗灰色区域而灰色区域也开始占据浅灰色部分。当我们完成测试执行时，大部分暗灰色和浅灰色区域将不复存在。

对于中间灰色的区域，在任何时间我们都可以生成一个详细的列表，包括测试过的风险，已知的通过测试的风险以及未发现任何必须修复缺陷的风险。对于浅灰色部分，我们可以生成一个详细的已知有测试失败并且有必须修复缺陷存在的风险列表。我们可以列出相关测试和缺陷。对于暗灰色部分，我们可以生成一个详细的测试未执行的风险



列表。这可以帮助我们准确地了解风险降低成果和剩余的风险级别。

让我们在接下来的基于风险测试的章节里对基于风险的测试是如何融入整个基本测试过程中的这个问题进行简单的描述。

在测试计划期间，我们将会执行包括风险识别和评估在内的质量风险分析。我们将根据风险来估算测试工作量和计划测试。

当继续剩余的测试过程时，我们将会调整质量风险分析并将其作为测试控制的一部分。

在测试分析和设计期间，我们根据风险级别来分配测试开发和执行工作。我们保持质量风险的可追溯性从而保证风险测试和正确测试排序的一致性，同时可以支持基于风险测试结果报告。

在测试实现和执行期间，我们根据风险级别对测试规程进行排序。我们使用探索性测试来检测未预料到的高风险区域，从而降低遗漏关键风险的几率。

在测试执行结束阶段评估测试退出准则以及报告测试结果的时候，我们将会度量风险的测试覆盖率并根据剩余风险报告测试结果（和发布是否准备就绪）。

最后，作为测试结束的一部分，我们需要在现场或生产中对系统行为进行评审。我们应当检查曾经发现和没有发现失效的地方，还应当检查那些特别重要和不那么重要的失效部分，从而看出质量风险分析是否准确无误。从这当中学到的经验教训可以帮助我们改进后续的质量风险评估准确度。

### 3.2.36 基于风险的测试与失效模式和影响分析练习 5

根据非正式风险分析和活动列表，设计基于风险的测试结果报告图表和/或表格来表达风险。

简单描述当项目在良好的情况下结束时这些报告的样子。

简单描述当项目在糟糕的情况下结束时这些报告的样子。

如果在教室里学习，请将大家分成 3~5 个小组。如果是单独学习，您需要独自完成这个练习。如果在教室里学习，当每一个小组完成了他们的分析后，对分析结果进行讨论。

我建议用 60 分钟来完成这个练习，包括讨论的时间。

### 3.2.37 基于风险的测试与失效模式和影响分析练习 5 参考答案

我倾向于用饼图的方法来显示剩余风险，至少在读者为高级管理层时是如此。详细的表格或柱状图可以按照质量分类来显示风险，这对于一线管理人员和独立的工作人员来说是有帮助的。

为了方便起见，让我们重述一下图 3-6 中的饼图。中间灰色部分代表所有相关测试被运行通过且没有发现必须修复缺陷的风险。浅灰色部分代表至少有一项相关测试没有通过且至少有一个必须修复缺陷的风险。暗灰色部分则代表剩下的那些没有已知的必须修复缺陷但测试还未完成的风险。

图 3-7 显示了测试执行前的饼图。如你所见，因为我们还没有运行任何测试也没有发现任何必须修复的缺陷，所以整个图都是暗灰色的。

图 3-8 显示了测试执行途中的饼图形状。有些风险项还未被覆盖到,大约 2/3 的部分已经被一个或多个测试覆盖到(记住:高风险级别的风险项很可能会被按照风险顺序运行的多项测试所覆盖,这就是暗灰色部分所表示的剩余未覆盖风险并未达到风险总数 50%的原因。)。因为此时许多缺陷和测试仍然没有结束,所以大约一半的部分是浅灰色的。但是如果按照风险顺序运行测试和修复缺陷,那么最高级别的那些风险项应当会被包含在 20%的灰色区域中。

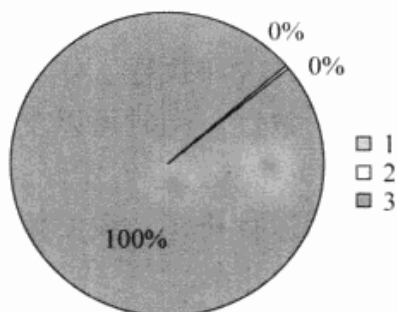


图 3-7 测试执行前的剩余风险图示

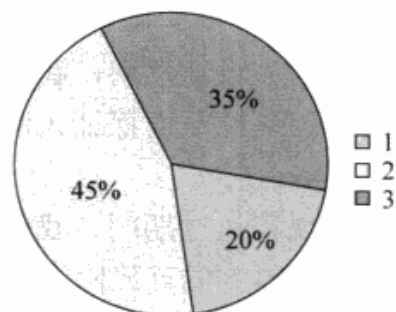


图 3-8 测试执行中途的剩余风险图示

当我们的测试进入项目尾声时,通常有 4 种情形,一种好的和 3 种不好的。让我们看一下这 4 种的所有情况。

图 3-9 显示了当我们即将成功完成项目时的饼图。我们的测试几乎已经覆盖了所有的风险项,而几乎所有这些测试都已经通过并且没有任何遗留问题。只有 2%的风险项有已知的必须修复缺陷或测试失效,另外只有 3%的风险项还没有进行测试。记住,如课程中提到的那样,我们可以在任何时间点上对风险项、失败的测试、浅灰色部分那些已知的缺陷和暗灰色部分那些未执行的测试进行细分。由于我们应当已经按照风险顺序运行测试和修复缺陷,所以在暗灰色和浅灰色的细分部分只会包含风险最低的风险项、测试和缺陷。假如管理层接受了这些浅灰色和暗灰色细分部分的已识别风险项,我们就可以进行发布了。

图 3-10 显示了当我们接近项目结束时仍有测试受困时的饼图。中间灰色的部分和前面测试执行中时候的变化不大。暗灰色和浅灰色部分也基本保持不变。浅灰色和暗灰色的细分部分会包含令人害怕的风险项、测试失败和已知缺陷。许多情况可能造成这种结果,例如缺乏变更管理导致没完没了的需求特征,不稳定的设计或者编码导致不断地引入新的缺陷,测试人员流失或测试环境损坏,或是开发团队修复缺陷失败。

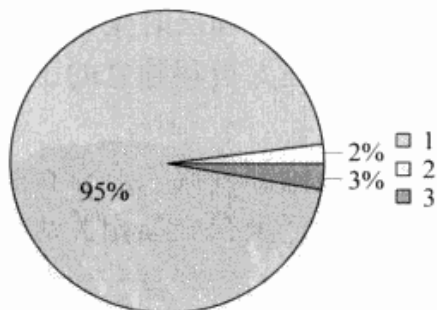


图 3-9 接近成功完成的剩余风险图示

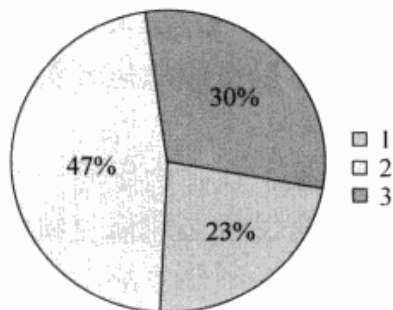


图 3-10 项目末期有测试受困时的剩余风险图示



图 3-11 显示了当我们接近项目结束时仍有大量未通过的测试用例和积压的缺陷存在时的饼图。灰色部分仍旧很小，而暗灰色部分可以忽略，超过 3/4 的风险项存在已知的必须修复缺陷或失败的测试用例。浅灰色的细分部分会包含大量的风险项、未通过测试和已知缺陷，而且其中许多都是相当严重的。许多情况可能造成这种结果，例如不稳定的设计或者编码导致不断地引入新的缺陷，或是开发团队修复缺陷失败。

图 3-12 显示了当我们接近项目结束时发现有测试倒退情况存在时的饼图。在这种情况下，我们可以看到暗灰色和浅灰色部分相比灰色部分增长得更多。暗灰色和浅灰色区域占了大部分。浅灰色和暗灰色区域的细分部分会带来大量的风险项、未通过测试和已知缺陷，而从项目中段开始，这个问题列表将不会变短反而会变得更长。许多情况可能造成这种结果，例如缺乏变更管理导致没完没了的需求特征，不稳定的设计或者编码导致不断地引入新的缺陷，开发团队修复缺陷失败，或是很晚才发现最初的质量风险分析中有许多遗漏的风险项从而导致测试范围显著增加。

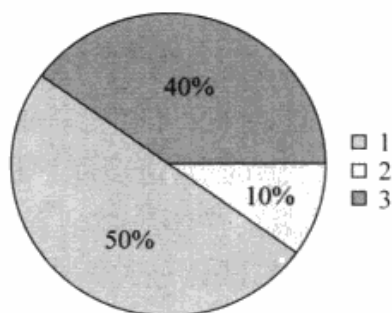


图 3-11 项目末期仍有大量积压的缺陷时的剩余风险图示

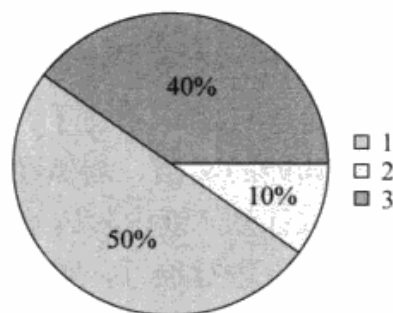


图 3-12 项目末期有测试倒退时的剩余风险图示

### 3.3 测试管理文档和测试计划文档模板

#### 学习目标：

(K4) 遵照 IEEE 829 概述测试管理文档，包括测试计划，测试设计文档和测试规程文档。

(K2) 指出至少 4 项包含在 IEEE 829 中的重要测试策略或者方法。

(K2) 阐述为什么测试管理文档能够管理测试策略中的偏差以及测试管理文档如何管理这些在测试策略中的偏差。

(K2) 概括 IEEE 829 中主测试计划的结构。

(K2) 解释说明 IEEE 829 所建议的测试计划的标准结构以及其关于根据组织、产品风险、项目风险、项目大小和项目形式采取的调整。

正确地管理测试当然不仅仅包括创建文档，通常来说在测试过程中进行结构化是必须的，而这通常包含文档。创建测试管理文档是一项普通的工作职责，虽然可能有些人称之为杂务，但是像我们这样的测试专家会发现有时很难对我们所创建的文档进行沟通。每个人都会或多或少地给这些文档起一些不同的名字，并且有时候同样的名字在不同的

组织中含义也不尽相同。

在 ISTQB 高级大纲中,我们试图通过定义 4 个关键的管理文档类型来对以上情形进行结构化。我们相信当组织遵照测试最佳实践的时候会遇到这 4 种管理文档类型。

**测试方针:**指描述组织哲学、组织目标和测试关键标准(很可能还包括质量保证的关键标准)的文档。

**测试策略(或称测试手册):**指描述组织通用测试方法的文档。理想状态下,该文档独立于任何一个特定的项目,仅仅涉及测试大部分项目的方法。其中包含产品风险管理、项目风险管理;组织的测试步骤、测试级别或阶段;在测试中的主要活动,例如那些在 ISTQB 初级测试过程中所描述的活动。你可能发现对这个文档的名字有些混淆,因为我们也从一系列组织测试原则的意义上讨论测试策略,例如基于风险分析的测试策略和动态的测试策略。注意,这个文档将那些组织原则详尽说明成为一个通用的可操作的方法。

**主测试计划(或称项目测试计划/测试方法):**指描述应用于特定项目的测试策略的文档。其中包括将发生的确切的测试级别,这些级别之间的关系以及如何管理和避免这些级别间的缺口和重叠。它也描述例如缺陷报告跨级别的活动,除非那些活动已经在测试策略中被足够详细地阐述了。注意,因为不同的团队可能拥有不同的测试级别,这可能是一个跨功能的文档。测试经理在创建这样的文档时应当明智地跨功能地进行工作。

**级别测试计划(或称阶段测试计划):**指为每一个测试级别描述具体活动的文档。它主测试计划上进行展开,明确已有的测试级别,例如系统测试计划、集成测试计划等。

如果你没有这 4 种文档来确切描述相关内容,是否你就不是一个好的测试经理呢?答案是否定的。你可以合并这些文档成为一个文档或者将其中的信息分散到多个文档中。在一些情况下,你可能根本不需要将这些信息文档化,虽然这是一个高风险的方法,但是对于那些非常小的、项目周期非常短的、临时性的项目来说仍然是可以使用的。

### 3.3.1 测试方针文档

测试方针描述了“为什么”一个组织要独立于方法、项目和测试级别进行测试。测试方针描述了组织的测试哲学。在一些情况下,它甚至更加广义地描述了贯穿软件生命周期的质量保证哲学。测试方针应该反映组织的价值和目标,这些价值和目标关系到测试和质量保证。

应由公司管理层而非测试团队来建立测试方针。然而,测试团队或者一些其他在信息技术团队、研发团队或者产品开发部门中的团队可能草拟测试方针并且提交管理层批准、颁布和建立。和任何建立方向的方针一样,测试方针应当被文档化。

测试方针可以是更宽泛的质量方针的组成部分。另外组织也可以选择编写和颁布完全分开的质量方针和测试方针。但是在这种情况下这两个文档应当是很好地互补和统一风格。

测试方针文档不应该是一份很冗长的文档。最好是简短的一两页内容。它应该是高度概括的文档,简明扼要地阐述以下内容:

- 测试目标,例如对系统建立信心,发现缺陷,提供信息和管理质量风险。
- 组织级基本的、通用的基本的测试过程,基于 ISTQB 的过程或者其他一些过程。



- 各类用来度量测试有效性和效率以及用来度量先前定义的目标的标准。例如，根据缺陷检测，测试方针可能提出测试中的缺陷检测率作为有效性的度量，然后对相关的测试中的缺陷发现成本和发布之后的缺陷发现成本进行比较作为效率的度量。
- 质量目标，如果文档中提到了质量保证。
- 测试过程中度量驱动的方法和基于经验进行改进的方法。例如在项目回顾中检查有效性和效率的度量结果然后通过实施关键测试过程、测试过程改进或者测试成熟度模型进行改进。
- 大型和小型项目新的开发和维护。

这个文档的读者不仅仅是测试团队，也不仅仅是技术人员，应该是包括以管理层为首的整个组织的所有人。就产品研发组织来说，组织也可以选择向客户和用户开放测试方针文档。

作为一个 RBCS 的咨询师，我可以说测试方针文档经常是缺失的。测试方针文档的缺失会使得对测试团队所要完成的工作缺乏一致认同并且无法对其目标进行度量，这对于测试组织来说是一个现实的问题。

### 3.3.2 测试策略文档

测试策略描述了整个组织通用的测试方法。理想的测试策略不是针对某个特定项目的测试方法而是适用于组织中大部分项目的。测试策略文档可以覆盖组织在测试中用来管理产品风险和项目风险的可选方法。例如，哪些是我们在测试中尝试应对的典型产品风险？我们该如何应对？这些产品风险如何关联到高阶的测试目标？哪些典型的项目风险可能影响测试？我们应该如何管理这些风险？

测试策略文档也可以提供组织在一般项目中的各种测试级别或阶段上的细节。例如，它可以讨论各种测试级别的入口准则和退出准则。它可以描述单元测试、集成测试和系统测试是如何进行管理的。针对每个测试级别，它可以对将要执行的高阶测试活动进行展开。

因为测试策略是对测试方针中提出的“为什么”提供具体“如何”进行的描述，所以它必须和测试方针相一致。而且，它应当为每个测试级别和总体测试提供一般的测试需求。例如，在基于风险的测试方法里，应当明确在质量风险识别和评估阶段一般需要检查的质量风险区域。

测试策略不依赖于任何项目，它是跨项目的。那就是说，测试策略文档描述了项目中应用于所有测试或大部分测试的测试方法。它也是跨责任团队的，包括不同团队在所有测试级别中采用的方法。例如，它可以包括单元测试、集成测试和系统测试以及每个测试级别由哪个团队负责。

测试策略可以提供测试过程的各种细节，可能包含以下部分：

- 集成规程，例如自顶向下、自底向上、功能或者基于风险。
- 测试规格说明技术，包括在高级测试分析师姊妹卷中的测试设计概念和所需文档细节级别。

- 所需测试独立性，可根据不同测试级别有所变化。
- 任何在测试中要遵守的必需的和可选的标准。
- 各种不同类型测试所用的各种测试环境。
- 应用于各种类型测试的自动化测试。
- 对软件工作产品和测试工作产品的可重用性的需求以及方法，例如测试用例、测试数据和测试计划。
- 再测试和回归测试的方法，这部分可能和自动化测试部分有交叉。
- 为了测试控制和报告所用的执行和信息收集方法。
- 测试度量以及沟通的频率和方法。
- 如何进行事件管理，包括事件报告、事件优先级排定和事件管理。
- 如何对测试件进行配置管理。
- 执行测试小组和其他项目中的小组之间的联系以及各种交付物在这些小组间的流转。

当项目类似的时候这些内容会相当有用，因为类似的项目会降低必需的项目特定文档量。测试团队应当在每个项目中都参考测试策略，而不是在每个测试计划中都重新对这些问题编写测试策略文档。

很自然的，不同的组织甚至同一组织内不同的项目可能需要不同的测试策略。测试策略文档并非创建一种“同一方法全体适用”的测试方法，而是创建一种可以调整的方法。

测试策略可以包含短期和长期的需求。因为这些需求可能会变化，测试策略中应当包括它们可能变化的原因以及这些变化会如何影响测试。测试策略可以根据项目和应用程序进行变化。例如，一个安全关键的应用程序需要采用更严格、更彻底或者符合特定标准的策略。

测试策略文档经常会被遗漏。这意味着人们并没有对这里提到的一些重要的部分进行描述。或者这些部分在每个项目里被提到1次、2次、3次、4次甚至更多次，而每次人们都会做出不同的决定。在每个项目中进行理性和慎重地调整是对的，但是必须要反复地搞清楚各个测试的基本方面，因为如果没有人能把握全局性的测试策略会导致效率低下，在许多项目里，这还会降低测试的有效性。

让我们看一个测试策略文档的简单的例子。图3-13展示了大型综合系统集成测试策略中的一些关键摘录。在这个项目里，我们对通过同一广域网与电话客服中心联系在一起的交互式语音应答（IVR）服务器进行测试。给电话客服中心服务器及服务及IVR服务器及服务的整合带来了较大的风险。

我们想要尽早开始集成测试从而降低这些风险。因此，我们可以把这归为一种预防性的策略。

每个版本被称为一个“主干”。从BB0到BB5共有6个主干。在可能的范围内，在每个主干里都会按照风险的降序增加系统和服务。换句话说，我们应该从具有最高风险的系统和服务开始。



**目标**

立刻开始集成测试从而加速发现/去除集成缺陷并且因此减少集成风险。

**主要测试分析、设计和实现任务**

- 分析系统设计和风险来定义各种主干，包括服务器和服务的子集。从最高风险的主干（BB0）开始，一直到最低风险的主干（BB5）。
- 为所有主干系统定义和实施命名修改。
- 定义和实施缺陷追踪和解决过程。
- 定义和实施主干测试环境和支持。
- 定义测试周期持续时间。
- 设计和实施每个主干的集成测试，包括自动测试用具和测试数据。

图 3-13 测试计划里的测试策略举例

每个主干包括前一个主干的系统和服务。因此，这是在基础大纲中提到的一种增量的集成策略。

注意测试策略是基于系统设计和相关风险的分析，因此，它是一种分析式的测试策略。

你可以想象得到，测试环境是复杂且不断变化的。因此，具备技能的支持队伍是必需的。

### 3.3.3 测试策略类型

有许多不同的测试策略，有些是文档化的，而有些只是口头惯例或者公司内部由来已久的理解，我们可以对各种测试策略类型进行了解。了解这些类型可以帮助你更清晰地知道如何对组织的测试策略进行选择、组合和设计。

你应当记得在基础级大纲里我们把测试策略分为预防性和反应式的策略。预防性测试策略专注于测试团队的早期介入和运用测试计划、分析和设计活动来识别项目计划、产品需求和设计以及其他项目工作产品里的问题。这使组织可以在测试执行前解决这些问题，从而防止在项目晚期发现大量缺陷，而在项目晚期发现大量缺陷是导致项目失败的主要原因之一。

反应式测试策略专注于当测试团队真正遇到问题时对系统做出反应。这些策略使组织在实际系统和起初计划的有所不同时可以避免测试工作产品开发相关的损失。

你可以看到，预防性的策略对那些可以预料的、小心计划的和按计划执行的项目是有优势的。如果你可以在项目里更早地进行各类测试活动，那么为什么要在测试执行的同时执行所有的测试计划、分析和设计呢？

对于那些混乱的、没有很好计划的或基本内容不断变更的项目而言，反应式的策略更有优势。如果当发布的系统被证明是面目全非而我们必须维护或者完全重建测试工作产品时，为什么还要创建那么一大堆测试工作产品呢？

这种把测试策略分为预防性和反应式的策略分类是一种简化的情况。在实际情况下，特定的测试策略会介于两者之间而经常同时吸收了预防性和反应式的特点。

在我的书“Pragmatic Software Testing”里，我调查了在实际使用中我们发现的测试

策略类型。顺便说一句，基础级大纲和高级大纲里测试策略列表正是基于这项调查。<sup>10</sup>让我们看一下现有的各种测试策略类型。

在例如基于风险的测试这样的分析式的策略里，测试团队对测试基准进行分析进而识别出测试条件。分析的结果形成了测试投入的基础。分析式的测试策略倾向于进行彻底的和完美的质量风险管理并且擅长发现缺陷。分析式的测试策略主要是预防性的并且需要大量的项目早期的时间和投入。

在基于模型的策略里，我们根据系统行为模型设计、实现和执行测试，例如那些依赖于运行概况的策略。如果模型捕捉到了系统的主要元素，测试将会令人满意。因此基于模型的策略依赖于测试人员开发良好模型的能力。如果这些模型或测试人员构建的模型无法捕捉到系统的核心或具有风险的部分，那么这些基于模型的策略将会失败。

在系统的策略里，我们主要遵照一些测试目标的标准目录，例如那些遵照质量特性检查列表的策略。系统的策略对于稳定的系统和那些与先前测试过的系统类似的系统来说是轻量而有效的。但是，当发生重大的变更时，除非至少对测试目标目录进行一些调整，否则将会使这些策略变得无效。

在过程一致或标准一致的策略里，我们主要遵照一些由标准委员会或某些睿智的人组成的团体所颁布的过程或标准，例如基于 IEEE 829 的策略。这些策略可以节省下您设计自己方法的时间和精力。但是，如果总体的测试任务和标准建立者知道的情况有所不同，或如果您遇到的测试问题和标准建立者已经解决的问题有所不同，那么您会发现这种借来的方法就像借来的燕尾服一样不适用。

在动态的或启发式的策略里，可以使用软件失效的通用规则；列出重要的软件部分、特征和行为；或者列出通用软件数据结构和操作来做出有根据的推测集中测试，例如那些基于缺陷分类或软件攻击的策略。这些测试策略将会把结构最小化而把灵活性最大化，同时通常这些策略专注于寻找缺陷（而不是为了建立信心或降低风险）。结构和文档的缺乏意味着缺少详细的覆盖率信息，没有系统的降低质量风险或包含任何预防性的元素。除了这些缺点以外，使用纯粹的动态策略要优于完全不进行测试。如果可以组合动态策略和分析式的策略，从而提供一种好的方法来弥补分析式的设计的测试中的不足那就更好了。

在咨询式的策略里，测试经理相信一些其他团队的人最了解什么应该测试，例如那些由用户或程序员决定测试内容的策略。测试经理询问这些人有关需要覆盖的测试条件并且在测试里覆盖这些条件。其他团队也担当主要的测试准则来决定期望结果。当测试团队主要作为其他团队的手和眼的时候，这种策略是适用的，例如有些类型的外包测试。但是通常来讲，这是一种低劣的专业测试替代品。

最后，在回归测试策略里，我们专注于反复测试和聪明的选择测试来尝试把对已工作部分影响的风险最小化，例如那些依赖于大规模功能测试自动化的策略。对于那些只

---

10 特别要提到的是，我在新闻组里和 Ross Collard、Cem Kaner、Kathy Iberle 以及其他进行了讨论从而得出了我的测试策略类型列表。我在“Pragmatic Software Testing”的课程材料和源自于“Pragmatic Software Testing”里用到了这个列表，同时在基础和高级大纲里也用到了该列表。



是慢慢变化的稳定系统，这些策略是有意义的。但是，对新特征没有进行良好的测试的风险总是存在的，而且如果变更的频率加快，那么对最近新增加特征的测试覆盖率将会无法跟上。除此之外，即使是一种好的测试自动化方法，快速增加的自动化回归测试集仍然会使得测试团队无法运行和分析所有的测试结果。

正如把测试策略分为预防性和反应式那样，记住它们是连续的统一体。另外还要记住策略是无关哲学或者宗教的，不需要仅仅选择一项而且永远忠实于它。策略是可以组合的。作为一个测试经理，用一些时间来仔细地选择和调整你的测试策略是一种明智的做法。可以让你的测试团队采用通用的测试策略或是针对某个特别的项目修改或完全重新制定策略。

### 3.3.4 测试计划模板

基础级大纲和高级大纲都对 IEEE 829 标准测试文档部分进行了讨论。在这个标准里有一些有用的测试计划和其他相关文档的模板。所以让我们看一下这些模板，但是在我们的讨论确切的模板之前，先让我们讨论一下如何聪明地使用模板。

主要有两种风险和使用测试模板有关（和其他任何目的的模板）。首先是人们可能把使用模板作为不再进行思考的借口。换句话说，他们专注于在模板的空白部分填空而不是完成测试计划、测试设计或测试实现。聪明地使用模板的方法应该是鼓励思考而不是妨碍思考。

第二种风险是“同一方法全体适用”的问题，这是指组织并没有针对他们的特殊需要正确地调整模板或项目团队并没有针对他们的项目进一步调整模板。这个问题可能导致在模板中忽略一些关键内容或者包含了一些不重要的内容。测试计划应当是计划活动而不是文档活动。

测试计划和其他测试文档在被正确地使用时，其内容和结构是会变化的。不同的组织会有不同的做法。内部文档标准将会影响到模板和文档。项目的风险、正式程度和规范会决定文档的类型和级别。

因为模板有利于增强稳定性和沟通，所以它对建立跨组织的通用文档结构有帮助。建立通用组织模板是有帮助的。聪明的使用者可以用这些模板来激励大家思考测试的问题。

一份测试计划是针对项目中整个或部分测试内容的子项目计划。一会儿我们将会看一下 IEEE 829 测试计划模板。可以将 IEEE 829 模板改写为每个具体的测试计划或是主测试计划。如果不需要遵照 IEEE 829，也可以创建自己的模板或纲要。而 IEEE 829 模板可以在您自己创建模板的时候提供一些有用的灵感。

您不应该实际上也不可能脱离项目的现实情况，完全从零开始编写测试计划。对测试进行计划的活动和测试计划文档都会被各种因素所影响。这些因素包括测试组织的测试方针；测试范围；测试目标和项目目标；项目风险和质量风险；功能、进度和预算相关的限制；被测试系统的关键区域；被测试系统的可测性以及资源的可用程度。同时，测试计划在被正确地使用时，它本身也是影响以上因素的一种工具。

一份 IEEE 829 测试计划包括以下部分：

- 测试计划标识符。
- 简介。
- 测试项（交付测试的一切内容）。
- 需要测试的功能。
- 不需要测试的功能。
- 方法（包括策略、组织的问题和测试范围）。
- 测试项通过/失败准则（我认为相当奇怪的一个部分，因为我认为不是测试项而是功能或者属性才可以通过或者失败）。
- 测试准则（例如，入口准则、退出准则、暂停和恢复准则）。
- 测试交付物（例如，报告和图表等）。
- 测试任务（我通常把这部分内容和“进度”部分进行合并而且列出关键的测试里程碑）。
- 环境需求。
- 责任。
- 人员和培训需要。
- 进度。
- 风险和应急方案（可以包括质量【或产品】和项目风险，虽然我经常将这部分内容分开在不同的部分进行讨论）。
- 批准（如果你需要签字审核）。

测试计划标识符  
简介  
测试项  
需要测试的功能  
不需要测试的功能  
方法  
测试项通过/失败准则  
测试准则  
测试交付物  
测试任务  
环境需求  
责任  
人员和培训需要  
进度  
风险和应急方案  
批准

图 3-14 IEEE 829 测试计划模板

为了便于参考，图 3-14 中列出了模板中主要的头部部分。

### 3.3.5 IEEE 829 软件测试文档标准以及它们如何与测试计划文档相关联

通常在创建测试设计、实现和执行过程中有 4 种其他的模板与测试计划紧密相连。每一份模板都覆盖了测试计划中讨论的某个部分并且进行了细化。当然，我们希望能够确保一致性。

第一份模板是 IEEE 829 测试设计规格说明。测试设计规格说明从高阶级别对测试用例和测试条件的集合进行了描述。一组测试集有时又被称为测试套件。IEEE 829 测试设计规格说明模板包括以下部分：

- 测试设计规格说明标识符。
- 要测试的功能（在这部分测试套件里指的是测试计划中要测试功能的子集）。
- 方法细化（对测试计划中描述的方法进行细化，讨论这部分测试套件的特定技术和工具等）。
- 测试标识符（用于在套件中追溯到测试用例）。
- 功能通过/失败标准（例如，测试准则、测试依据和遗留系统等）。



为了便于参考，图 3-15 中列出了模板中主要的头部部分。

测试套件和套件内的测试用例经常按照风险和业务优先级进行排序。而测试计划中往往会讨论到这些优先级。项目限制、资源和进展会影响到排序，而这部分内容也包含在测试计划中。

接下来第二份模板是 IEEE 829 测试用例规格说明。测试用例规格说明对测试用例的细节进行了描述。该模板包括了以下部分：

- 测试用例规格说明标识符。
- 测试项（交付测试的内容，这里指的是测试计划中测试项的子集）。
- 输入规格说明（用户输入和文件等）。
- 输出规格说明（期望结果，包括界面、文件和时间等）。
- 环境需求（硬件、软件、人员和支持，这里指的是测试计划环境需求的子集）。
- 特殊程序需求（例如操作者干预或许可）。
- 用例间依赖性（必要的话需要设计前置条件）。

虽然在这个模板里定义了内容的标准，但是测试用例的具体内容仍然是开放特定的。在实际情况下，测试用例在投入、持续时间和覆盖的测试条件数量方面会有重大的变化。测试计划应当包括这些问题。

为了便于参考，图 3-16 中列出了模板中主要的头部部分。

测试设计规格说明标识符 要测试的功能 方法细化 测试标识符 功能通过/失败标准
---

图 3-15 IEEE 829 测试设计规格说明

测试用例规格说明标识符 测试项 输入规格说明 输出规格说明 环境需求 特殊程序需求 用例间依赖性
--

图 3-16 IEEE 829 测试用例规格说明

接下来讨论 IEEE 829 测试规程规格说明。测试规程规格说明描述了如何运行一项或多项测试用例。这个模板包括以下几部分：

- 测试规程规格说明标识符。
- 目标（例如需要运行哪些测试）。
- 特别需求（技能、许可和环境等）。
- 规程步骤（日志、设置、开始、前进【步骤本身】、结果度量、关闭/暂停、重启【如果需要的话】、停止、包装/拆卸、紧急情况）。

为了便于参考，图 3-17 中列出了模板中主要的头部部分。

虽然 IEEE 829 标准中将测试规程和测试用例加以区别，但是在实际工作中测试规程经常是嵌入在测试用例中的。换句话说，它们同样描述的是采取的行动、输入和其他使用的数据以及验证的期望结果的文档。但是，对于自动化测试而言，区分脚本和输入以及期望结果是很重要的，我们将会在后面的课程中讨论这部分内容。

有时候测试规程指的是测试脚本。测试脚本可以是手工或者自动化的。

IEEE 829 标准里的另一个与测试计划相关的文档模板是测试项传递报告。在 IEEE 829 模板里这份文档是很独特的，因为经常不是测试人员而是其他人来创建这份文档。通常发布工程团队或者同等的团队将会创建该文档并将其及测试提交测试人员。

很自然的，测试项传递报告描述了交付测试的测试项。IEEE 829 测试项移交报告模板包括以下部分：

- 测试项传递报告标识符。
- 传递项（包括传递项名字和修订号，这可以追溯到测试计划中的测试项）。
- 测试对象或测试项的位置，包括它们在哪里，它们从哪里来，发布媒介是什么，如何进行标号以及其他为了清楚识别测试环境里的测试对象或测试项所需的信息。
- 测试对象或测试项的状态，包括修复的缺陷和引入的变更等。
- 最后，如果有对测试对象和测试项进行变更控制，记录下批准发布测试对象或测试项进行测试的人的名字。

为了便于参考，图 3-18 中列出了模板中主要的头部部分。

测试项传递报告有时候也被称为发布说明。发布说明经常只包括上述内容中的部分信息。它们经常是非正式的文档。测试计划应当说明发布工程团队必须提供交付的测试项内容和交付方式等信息。否则，您很可能会发现无法得到所需信息甚至无法得到正确的测试项。

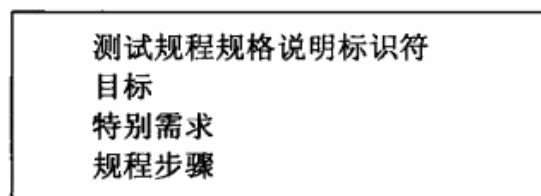


图 3-17 IEEE 829 测试规程规格说明

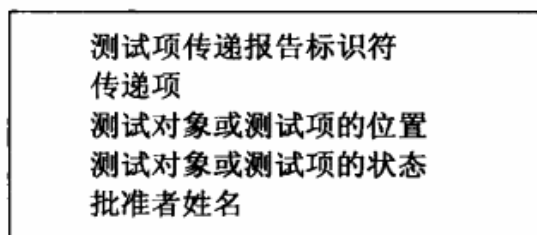


图 3-18 IEEE 829 测试项传递报告

因为 IEEE 标准包括了许多相互关联的文档，让我们大致浏览一下这些文档。建议您先花一点时间学习一下图 3-19，然后再看后文中对图 3-19 的解释。

在图 3-19 的中间部分，我已经画了项目的时间轴。从项目初始阶段开始，接着是计划阶段，然后是分析、设计和实现阶段，再然后是测试执行和控制阶段，最后是测试结束阶段。这条时间轴包括了第 2 章里提到的基本测试过程，它把 IEEE 829 文档分布到了基于 ISTQB 基本测试过程进行测试的典型项目当中。您可以看到在图的左边部分都标注了角色，从而和右边部分的工作产品相关联。

### ISTQB 术语

级别测试计划：通常用于一个测试级别的测试计划。





当测试人员运行测试的时候，他们会在测试日志里记录下测试结果。测试日志记录了运行的测试规程。在运行测试过程中，测试人员通常会报告事件。在给定的测试执行期间会生成许多测试事件报告。测试事件报告会被关联到特定的要测试的测试对象。测

试事件报告也会根据期望结果被关联到测试用例规格说明。测试事件报告按照运行的测试规程关联到测试规程规格说明。在实施某些标准时，它也会被关联到测试用例。当然在测试规程和测试用例是同一文档的情况下，这个关联对两者都有效。

在测试执行期间，测试经理和测试分析师不仅会创建测试日志，还会生成一份或多份测试总结报告。测试日志里的信息会关联到测试总结报告。测试总结报告也会按照测试项、测试项通过/失败准则和计划的和实际的测试任务关联到测试计划。在测试结束期间，测试经理可以生成最终的测试总结报告来总结整个测试执行阶段的结果。

您可以预料到 IEEE 829 文档标准在 ISTQB 高级考试中是很重要的。推荐您回顾图 3-19 直到弄清楚整个标准为止。

### 3.3.6 主测试计划文档

清楚地了解了 IEEE 829 测试计划模板和相关模板以后，让我们进一步讨论主测试计划和级别测试计划。主测试计划描述了如何在项目里跨多个测试级别地进行测试。它包括了描述如何在特定的项目中应用测试策略。当测试被分解为多个级别时，例如单元测试、集成测试和系统测试，主测试计划应当描述这些级别间的关系以及工作分配。

当编写主测试计划时，应当使其与测试方针和测试策略保持一致。但是在有些情况下，主测试计划会因为特定的原因和测试方针和测试策略相背离。您应该在主测试计划中解释其原因。

我们想要保证主测试计划在进行测试的项目或运营中作为项目计划或运营说明的补充。它应该描述大型项目或运营中的测试部分。它应该保证测试是按照服务于项目或运营用途和目标以及更广的组织测试目标来进行组织的。

如前面提到的那样，主测试计划的特定内容和结构可能会发生变化。但是，我看到的和参与编写的大部分好的主测试计划都倾向于包括 IEEE 829 模板里的那些主题，虽然经常会用不同的名字。

对测试计划中的测试工作范围进行说明是十分重要的，因此我喜欢看到有一部分文档是根据要测试项和不要测试项以及要测试质量属性和不要测试质量属性来描述这方面内容的。测试计划应当提及测试进度和预算以及那些受限于项目或运营预算的内容。当项目包含多个测试级别时，确保在每个测试级别上都对范围进行清晰定义。

因为在协调测试项发布、测试执行周期和最终的测试周期完成时经常会发生沟通错误和误解，测试计划应该在某种程度上具体描述测试执行周期及其和软件发布计划间的关系。在什么情况下我们可以开始一个级别的测试？在什么情况下我们可以暂停和恢复一个级别的测试？在什么情况下我们可以宣布一个级别的测试完成？通常在项目团队中进行传递的时候很可能会发生混乱，因此测试计划应当讨论将会在测试团队和其他人或部门间传递的关系和交付物。

测试计划应当直接或间接地提到组织的测试目标，这是进行测试工作的商务理由和价值。我们在测试中覆盖的产品风险有哪些？会影响到测试的项目风险有哪些？

在有些情况下，主测试计划和级别测试计划可以是同一份文档。这在只有一个测试级别是正式级别的时候很常见。例如，如果独立测试团队执行了正式系统测试，但是程



程序员执行了非正式的组件和集成测试，而客户在 Beta 测试中执行了非正式的验收测试，那么系统测试计划很可能是唯一的书面的测试计划。所以它将需要覆盖前面所提到的内容。

测试处在项目中其他所有活动的下游。这意味着如果我们不仔细的话，那么在项目其他地方的混乱和无序会渗入测试工作。如果在测试计划阶段识别到项目中其他任何地方的无序可能将会影响到测试时，主测试计划或级别测试计划需要弥补其他项目文档中的不足。例如，如果配置管理过程没有被文档化，你可能会在测试项发布、测试项传递报告、修订版本号和其他发布工程相关的部分遇到挑战。所以测试计划应当详细说明测试对象将会如何交付给测试团队。

### 3.3.7 级别测试计划文档

级别测试计划描述了在单个项目的单个测试级别上如何进行测试。级别测试计划会描述测试级别上的特定活动。理论上每个测试级别都会对应一份从主测试计划中该测试级别内容扩展而来的级别测试计划。级别测试计划提供了特定级别的进度、任务和里程碑细节。如果有特定级别的标准和模板，那么也会被包含在级别测试计划之内。

虽然理论上要求一份主测试计划和 4 份或更多的级别测试计划每份对应一个测试级别，但是在非正式项目或运营工作中你可能发现只有一份级别测试计划是书面的测试管理文档。如果是这样的话，这份文档里应该覆盖理论上应该在主测试计划中被覆盖的内容，并且甚至可能还要包括测试策略和测试方针中的内容。

### 3.3.8 测试计划和测试计划偏离案例学习

图 3-20 是一个类似 IEEE 829 的测试计划纲要示例，这个例子来自于已经提到过多次的交互式语音应答服务器项目。图中列出的是计划中的主要头部部分。

<ul style="list-style-type: none"> <li>● 简介</li> <li>● 范围、定义和背景 <ul style="list-style-type: none"> <li>● 范围</li> <li>● 定义</li> <li>● 背景</li> </ul> </li> <li>● 入口和出口准则 <ul style="list-style-type: none"> <li>● 预集成测试入口准则</li> <li>● 预集成测试停止准则</li> <li>● 预集成测试退出准则</li> <li>● 正式集成测试入口准则</li> <li>● 正式集成测试停止准则</li> <li>● 正式集成测试退出准则</li> </ul> </li> <li>● 质量风险</li> <li>● 里程碑进度计划</li> </ul>	<ul style="list-style-type: none"> <li>● 测试环境 <ul style="list-style-type: none"> <li>● 系统修订版本</li> <li>● 主干内容追踪</li> <li>● EPCOT 和实验室</li> </ul> </li> <li>● 测试开发</li> <li>● 测试执行 <ul style="list-style-type: none"> <li>● 人力资源</li> <li>● 测试硬件</li> <li>● 测试和问题追踪和管理</li> <li>● 问题隔离和分类</li> <li>● 测试周期</li> </ul> </li> <li>● 变更历史</li> <li>● 参考文档</li> </ul>
---	---

图 3-20 测试计划的主要部分示例

这个计划包括了两个测试级别：预集成测试和集成测试。在这个项目里，预集成测

试相当于 ISTQB 大纲和术语中定义的系统集成测试。集成测试相当于我在先前第 1 章里提到的综合系统的系统测试。

你可以在图中看到定义各种入口、停止和退出准则的部分。确保两个级别测试间的协调是很重要的。

还有其他级别的测试。正式地来讲,综合系统中包含的每个系统都有系统测试级别。这些系统还有非正式单元测试级别和组件集成测试级别,这部分内容属于开发团队。这些测试级别中的一部分最终也没有进行或者至少没有被彻底地执行,这会导致许多缺陷在项目晚期才被发现。

通常综合系统项目的测试环境都很复杂,这个项目也不例外。可以看到我们在应对这个问题上主要有 3 个方面:

- 系统修订版本部分描述了我们在测试执行期间组合实时测试环境,而这将最终成为生产环境(在某些情况下,它可能最终不是那么实时,一会儿我会提到这部分内容)。因此,哪些环境版本将会可用?在何时?测试计划将会在这个部分解答这些问题。
- 主干内容追踪指的是每个集成构建都包括哪些特定的软件和系统修订版本,在这里也被称为主干。因为项目经常会发生变更,需要确保我们知道要测什么。
- 最后,EPCOT 和实验室部分包括了基础设施和设备。EPCOT 是我们给走到一起的测试环境起的一个聪明的名字,这是因为我们希望这个名字能够告诉我们明天会怎样。

至今为止测试计划都是良好的,接下来的问题是项目最终会发生什么。追踪测试计划中的偏离是一个好主意。IEEE 829 标准建议使用测试日志来追踪导致测试计划中偏离的事件。在这个特定的项目里,我用了一个简单的 Access 数据库表格作为我的日志。图 3-21 是 27 页的报告中的其中一页,用来描述各种测试计划偏离的情况。我已经按惯例对一些信息进行了编辑来保护无辜者以及那些不那么无辜的人。

前两项是有关必须处理迫在眉睫的 Beta 硬件资源损失。Beta 是 IVR 服务器之一,它是符合要求的早期模型之一。还有两台被称为 alpha 和 baby alpha 的模型。但是我们并没有在测试中使用它们,因为它们并不具备所需的完全呼叫能力。

最后,你可以发现 Beta 并没有消失,但是它确实接受了升级。升级使得在测试活动中遇到了罕见的中断,因此它确实干扰了我们的测试工作。

第三项讨论了由 TIBCO 许可证问题引起的测试问题。TIBCO 在这个项目里提供了一些中间件。逻辑和配置问题导致了许可证问题,这对测试产生了负面影响。

当需要向项目经理和其他项目利益相关者解释可能发生并影响测试的事件时,这些不同种类的日志会非常有用。你可以使用 Word 文档,但是我建议使用 Access 数据库或 Excel 文件。Access 或 Excel 使你可以对影响测试的进度、资源和其他成本中的各种变更的影响进行计算。



影响	项目名称	记录	变更描述	影响描述	进度影响	资源影响	其他成本开支/节省
8/4/1998	集成	7/31/1998	我们将会失去 Beta, 它将会被给 XXX 使其能够完成他们的 QVS API 开发和测试工作	如果我们得到 CDR 作为协同主机来进行压力和能力测试, 影响应当会被最小化。如果没有, 那么这对我们来说是一个严重的问题。同时, 如果 baby-alpha 在关键路径上那么集成可能会受到负面影响			
8/4/1998	集成	8/11/1998	最后的或据说最后的 Beta 的计划是它将不会回到 XXX, 而是留在原处。OOC 会在 8/10 这周赶上 UUU 而使其达到生产就绪状态并且对操作单元的网络团队进行培训	实际上没有影响。Beta 服务器不能工作几天没有问题, 因为我们在 post-BB [0]/pre-BB[2] (非计划的) 中断中	0	0	\$0
8/10/1998	集成	9/10/1998	TIBCO 许可证频繁过期。从前是每小时, 后来是每天, 现在是每周	当这个问题发生时, 测试中断并被迫重启。这很可能耗费 20 个小时左右的测试时间以及可能 10 人一小时左右投入。这也会降低对测试结果的信心	3	13	

图 3-21 测试计划偏离举例

### 3.3.9 主测试计划案例学习

图 3-22 是一个类似 IEEE 829 的主测试计划纲要示例, 这次的项目是我已经提过多次的因特网应用程序项目。它描述了硬件和软件测试并用分开的部分描述了每个部分的质量风险。针对硬件测试有一份级别测试计划。主测试计划包括了软件测试的细节。

软件测试包括两个级别: 集成测试和系统测试。集成测试并不是很正式, 所以虽然我们对其设定了入口准则, 但是我们并没有设定继续准则或退出准则。集成测试会在所有的集成测试结束后结束。系统测试有正式的入口、继续和退出准则。

### 3.3.10 一个简单 PC 应用程序的测试计划案例学习

我们已经从前面的内容里了解了测试计划的框架和主要部分。接下去让我们更近距离地、有选择性地看一些与 IEEE 829 类似的测试计划。这个测试计划描述了对一个简单



<ul style="list-style-type: none"><li>● 简介</li><li>● 界限<ul style="list-style-type: none"><li>● 范围</li><li>● 定义</li><li>● 背景</li></ul></li><li>● 质量风险<ul style="list-style-type: none"><li>● 硬件</li><li>● 软件</li></ul></li><li>● 转换<ul style="list-style-type: none"><li>● 集成测试入口准则</li><li>● 系统测试入口准则</li><li>● 系统测试继续准则</li><li>● 系统测试退出准则</li></ul></li></ul>	<ul style="list-style-type: none"><li>● 进度</li><li>● 测试配置和环境</li><li>● 测试开发</li><li>● 测试执行<ul style="list-style-type: none"><li>● 人力资源</li><li>● 测试用例和缺陷追踪</li><li>● 缺陷隔离</li><li>● 发布管理</li><li>● 测试周期</li></ul></li><li>● 风险和突发事件</li><li>● 变更历史</li><li>● 参考文档</li></ul>
---	---

图 3-22 主测试计划示例

PC 应用程序的测试。这个应用程序是被设计用来在 PC 评估和采购过程中进行比较性的 PC 性能测试。它位于专用的 USB 驱动器。

RBCS 在这个项目里充当测试团队。我们在应用程序的用户接口上运行组件测试。这是一个相对短的、低成本的和低风险的测试项目，因此可以看到在测试计划中的正式程度和严格程度也就相对较低。和前面一样，在本案例学习中非 RBCS 的姓名已经被修改过了从而隐去客户和参与者的真实姓名。

第一部分摘选是有关资源的部分。在表 3-11 里我们定义了测试团队和客户联系人。考虑到这个项目的简单性，唯一的测试团队外部的联系人是 Quincy。

表 3-11 资源案例学习

职 位	角 色	姓 名
测试经理	计划、跟踪和报告测试执行。管理每日工作汇报、状态更新和向客户联系人及测试团队报告结果。确保合适的资源。审核缺陷报告和测试结果	Rex Black
主测试工程师	执行测试。指导其他测试工程师。在需要的时候担任测试经理角色。审核缺陷报告和测试结果	Nabi Berri
测试工程师	执行测试。在需要的时候将结果报告给主测试工程师或测试经理。审核缺陷报告和测试结果	Amit Harare Laurel Becker
测试系统管理员	确保对所有的测试环境进行正确的配置	Joe Maxim
客户联系人	交付测试发布。接收测试结果。为第二阶段的缺陷解决提供指导。对任何测试团队无法解决的阻碍测试的问题进行外部联络	Quincy Adams

图 3-23 显示了测试计划的测试执行过程部分。在我对其中的关键思想进行描述之前，建议您先完整地阅读一下这张图。

注意我的第一句话就是有关建立测试目标。这个项目唯一的目标就是发现缺陷。因为这是一个一次性的项目，因而不会有持续的测试团队。我把这个声明放在了测试计划里，否则这将会被放在测试方针文档里。



【测试项目】的目标是发现在【应用程序】用户接口中的缺陷。RBCS 测试团队倾向于在两个测试版本上通过运行一组测试章程和探索性测试来达到这个目标。在两个组件测试阶段各有一次测试发布提交给 RBCS。测试团队将会在两个组件测试阶段中的每个阶段正确地运行一遍所有的测试用例。

为了跟踪测试，测试团队将会使用……测试用例总结工作表……在每个测试周期开始的时候，测试经理……将会给每个测试人员分配一个任务。在第二阶段，测试经理……将会尝试给每个测试人员分配和第一阶段不同的任务来降低单个测试人员不正确的假设导致测试遗漏的风险。一旦任务被分配，这些测试人员将会遵照剩余部分所描述的过程来执行每个测试用例并且重复这个过程直到他们完成指派的工作为止。

如果他们在测试周期结束前完成了他们的工作，可以通过重新分配一些其他测试人员的测试用例给他们来协调工作，从而帮助其他的测试人员。

图 3-23 案例学习测试执行过程

我也在这里简要地定义了我们的测试策略，那就是综合使用咨询性测试（客户提供测试章程）、基于对象的分析式的测试（分析客户提供的用例来进行测试条件识别）和反应式的测试（使用探索性测试）。我们同时还在这部分内容里定义了发布接收、测试周期以及测试用例和测试周期的映射。

下一段描述了测试工作指派和跟踪过程以及测试人员运行测试的过程。注意这里也提及了工作量平衡的问题。

表 3-12 是一个客户提供的用例和测试章程示例。同样，我推荐在读完这个表之后再继续学习后面的内容。

表 3-12 案例学习用例和测试章程

名 称	测试当前电脑
用户目标	计算和查看插入【应用程序 USB 驱动器】的电脑真实性能评分
起始点	测试当前电脑可以从任何应用程序状态初始化
路径	用户在主要功能区域点击测试。动画运行在内容区域，与此同时执行【性能测试】并建立真实性能评分。真实性能评分一旦建立，动画立刻停止并且在内容区域显示当前电脑的真实性能评分
终点	当前电脑的真实性能评分在内容区域被显示出来
注释	即使当前电脑的这个场景已经被执行了一次或多次，点击测试仍然进行【性能测试】并且显示新的真实性能评分
测试章程	验证起始点到终点的路径，包括指定的错误条件

可以看到这里有许多机会来进行测试设计和探索性测试。例如，用例里说该用例可以从任何应用程序状态开始。那么有多少不同的状态呢？我们已经对所有的状态进行测试了吗？

还要注意测试章程的简要程度。这会给测试人员提供许多不同的方法来探索用例。

表 3-13 是项目里程碑的进度表。同样，我推荐在读完这个表之后再继续学习后面的内容。

表 3-13 案例学习里程碑进度表

日 期	里 程 碑
9 月 30 日	向 RBCS 提交测试章程
10 月 7 日	RBCS 提交 AMD DIPstick 测试项目的系统测试计划
10 月 13 日	向 RBCS 提交第一个测试发布（假设是功能完整的）
10 月 13 日	向 RBCS 提交所需的 shell DLL 文件来【创建测试数据】
10 月 14 日—10 月 20 日	RBCS 第一阶段测试
10 月 21 日	RBCS 提交第一阶段测试结果和缺陷报告
10 月 27 日	向 RBCS 提交第二个（最终的）测试发布
10 月 28 日—11 月 3 日	RBCS 第二阶段测试
11 月 4 日	RBCS 提交第二阶段测试结果和缺陷报告

注意用例和测试章程作为测试基础是在项目开始的时候提交的。我们对这些信息进行分析并开发出测试计划，然后我们有两个时长为一周的测试周期，在这两个测试周期之间有一个星期的间隔用来修复缺陷。整个项目仅耗时一个月多一点。

最后，我们在表 3-14 里看到了项目风险和紧急情况。对于这个简单的项目而言，风险也相当简单。需要注意的是在这个项目里我们并没有许多能力来控制风险。许多紧急情况会导致测试减少、测试延迟或两者同时发生。

表 3-14 案例学习风险和紧急情况

风 险	紧 急 情 况
在第一或第二测试阶段中测试发布的缺陷很多	运行所有能够运行的测试。允许在测试阶段中存在中间阶段的测试发布，完成尽可能多的测试
在第二阶段结束时组件中仍然存在许多缺陷	因为项目范围内没有提供额外的测试，所以这是一个未缓解的风险
第一阶段或第二阶段的开始时间被延迟	测试开始时间延迟。因为测试人员可用性和特定的进度日期紧密绑定，测试开始时间的延迟可能会导致结束时间不成比例地更长时间的延迟

### 3.3.11 测试管理文档和测试计划文档模板练习

回顾 HELLOCARMS 系统需求文档。用 IEEE 829 模板根据前面的风险分析练习信息和对 HELLOCARMS 项目的分析勾画一个包括系统测试和系统集成测试的组合的主测试计划和级别测试计划。使用一个具体解决方案级别，每个部分有 2~4 个解决方案。

如果在教室里学习，请将大家分成 3~5 个小组。如果是单独学习，您需要独自完成这个练习。如果在教室里学习，当每一个小组完成了他们的分析后，对分析结果进行讨论。

我建议用 90 分钟来完成这个练习，包括讨论的时间。

### 3.3.12 测试管理文档和测试计划文档模板练习参考答案

下面就是我为这个练习建立的提纲：



### 测试计划标识符

遵循 Globobank 现有的文档命名/编号标准。

### 简介

概要描述项目相关信息。

概要描述测试相关信息。

### 测试项

Globobank 系统包括以下组件：

- SQL 和其他数据库服务器部分。
- Java 和其他应用程序服务器部分。
- Java、HTML、Flash 和其他 Web 服务器部分。

Globobank 系统包括以下集成组件：

- LoDoPS 接口。
- Decisioning 主框架接口。
- 其他接口组件。

所有以上部分在提交给数据中心时必须用同样的方式打包。

### 需要测试的功能

房屋净值贷款、房屋净值信用贷款和反向住房按揭贷款功能。

拒绝不想要的申请者。

端到端事务处理。

错误处理和恢复。

安全性。

法规遵循。

HELLOCARMS/评分主框架接口。

HELLOCARMS/LoDoPS 接口。

HELLOCARMS/GLADS 接口。

HELLOCARMS/GloboRainBQW 接口。

### 不需要测试的功能

【在这个时候我们只识别功能质量特性】

### 方法

主要测试策略是分析式的基于风险的测试，这部分占 60% 的测试执行投入量，这些测试投入按照风险级别进行分配。

第二测试策略是反应式的策略，使用软件攻击和探索性测试，这部分占 20% 的测试执行投入量。

用确认测试来验证缺陷修复并检查先前失败的测试用例是否能成功通过，这部分占 20% 的测试执行投入量。

系统测试和系统集成测试将会由 HELLOCARMS 独立测试团队负责，假设一部分基本单元测试和组件集成测试由开发人员负责。

### 测试项通过/失败准则

我会根据度量的产品质量把这部分包含在测试准则部分。

### 测试准则（例如，入口、退出、暂停和继续准则）

需要根据以下几点为系统和系统集成测试级别定义入口准则：

- 测试件、测试环境和测试团队的准备情况。
- 测试项的质量（单元和组件集成测试结果、冒烟测试、构建验证测试等）。
- SDLC 生命周期模型规则。

需要根据以下几点为系统和系统集成测试级别定义退出准则：

- 用缺陷度量元评估产品质量。
- 用测试用例完成度量元评估测试完整性。
- 用测试时间度量元评估测试效率。
- 用测试覆盖率度量元评估测试彻底性。
- 剩余风险级别。
- 已知缺陷解决方案。
- 管理层接受的剩余风险。

### 测试交付物

定义 4 个或 5 个在测试执行期间提交项目经理的测试结果图形化报表和支持数据。

定义在测试执行期间提交技术支持和开发部门等的具体信息。

定义将会在测试执行结束时提交维护测试的测试件、测试环境和测试过程等。

### 测试任务

参考测试估算工作分解结构。

### 环境需求

数据中心和呼叫中心的复本（尽可能接近），在 HELLOCARMS 系统需求文档的图 1 中有提及。

与评分系统、LoDoPs、GLADS 和 GloboRainBQW 的测试接口，对数据交换进行足够的控制从而生成所有必要的输入和输出。

### 责任

定义测试团队成员和他们的角色。

定义主要的非测试参与者。

### 员工和培训需求

定义团队，可以参考前面的部分。

如果需要，讨论技术、测试或业务领域的培训。

### 进度

参考测试估算工作分解结构。

### 风险和紧急情况

参考质量风险的质量风险分析文档。

参考质量风险练习里识别的项目风险。



批准

如果需要，在这里增加一个签名批准的部分。

## 3.4 测试估算

### 学习目标:

(K3) 考虑影响成本、工作量和时间的因素，以一个小型样例项目进行基于度量元或基于经验的方法估算测试工作量。

(K2) 理解并举例说明在大纲中提到的可能引起估算误差的因素。

估算是一种管理活动，它经常是计划活动的一部分或前序。但是估算也可能是销售或投标活动的一部分。估算生成任务的约计成本和进度目标，这是估算者预料会在一个特定项目或一些特定的持续操作中发生的内容。

一个好的估算在生成时有许多特性：

- 它基于有经验的从业者的知识和智慧。
- 它获得了那些将参与工作的人的支持。
- 它提供一系列明确的和详细的内容，包括资金、资源、任务和相关人员。
- 它基于每个任务的最可能的成本、工作量和持续时间。

假如一个估算提前有了这些特性，那么它也很有可能在项目完成之后拥有最重要的特性，那就是对成本、资源和所需时间准确地预测。

不幸的是软件和系统工程的估算是很困难的，我们的估算很少能达到上面提到的重要特性。这个不仅是技术问题，而且还是政治问题（也许这是大部分原因）。赢得对现实的、准确的和完整的估算的认同是困难的。这也就是说，公认的项目估算最佳实践，应用这些最佳实践将会帮助你更好地进行估算。

在测试估算里，我们应用已知的最佳实践对项目 and 持续进行的测试工作进行估算。测试估算应当包括所有测试过程中的活动。例如，可以运用 ISTQB 基本测试过程来生成工作结构分解（经常通过甘特图来展现）。这可以包括以下主要活动：

- 计划。
- 控制。
- 分析。
- 设计。
- 实现。
- 执行。
- 结果报告和退出评估。
- 结束。

在这些活动中，测试执行受到管理层的关注最多。因为测试执行处于关键路径上，所以其成本、工作量和持续时间很关键。在被测试系统的质量低下或未知的情况下，测试执行估算是相当困难和不可靠的，不幸的是我们经常遇到这样的情况。测试执行所需

时间的主要决定因素经常不是运行所有测试用例一次所需的时间，而是发现、修复和确认修复所有的关键缺陷所需的时间。

在某些情况下，人们尝试通过估算测试用例的数量、工作量和持续时间来对测试执行进行估算。但是，除非你能够有一种根据发现/修复/确认周期对估算进行调整的方法，否则这种估算可能会导致大家基于低测试用例失效率的假设来进行测试估算。而这通常不是一种好的假设。

这也就是说，我们将会在估算中进行一些假设。应当确保你把所有重要的估算假设都文档化并且在测试控制中仔细观察这些假设可能失效的信号。如果你可以把这些假设放在工作表或类似微软 Project 那样的项目管理工具中进行管理，那么你就可以随着项目的推进重新估算进度并且对原估算进行修改。

### 3.4.1 影响估算的因素

当你在估算中开始考虑这些假设条件时，经常会了解可能影响测试活动成本、工作量和持续时间的因素，考虑所有这些因素是很重要的。

我认为这些因素通常分为以下 4 类：

- 过程因素。
- 物质因素。
- 人为因素。
- 延迟因素。

让我们分别来看一下这 4 类因素。我会从让测试估算更快、更便宜和更准确可靠的角度来分别回顾过程、物质和人为因素。而延迟因素则更倾向于让较慢速的、较昂贵的测试和不精确不可靠的估算联系在一起。

一些在工作过程中产生的影响估算的过程因素：

- 从项目的第一天起，项目中测试活动的范围。
- 清晰地定义测试组织和其他组织间的信息传递方式。
- 对项目、测试计划、产品需求、设计、实施和测试有管理良好的变更控制过程。
- 所选定的系统开发或维护生命周期，包括生命周期中的测试和项目过程成熟度。
- 及时和可靠的缺陷修复。
- 现实可行和可操作的项目以及测试进度和预算。
- 及时地提交高质量的测试交付物。
- 早期测试阶段的正确执行（单元、组件和集成）。

一些由于项目性质、所用工具和可用资源等产生的影响估算的物质因素：

- 现有已消化的高质量测试和过程自动化以及工具。
- 测试系统质量，包括测试环境、测试过程、测试用例、测试工具等。
- 足够的、专用的和安全的测试环境。
- 单独的、足够的开发调试环境。
- 可靠的测试准则的可用性（因此我们看到一个缺陷时可以知道它是缺陷）。



- 可用的、高质量的（清晰、简明、准确等）项目文档，例如需求、设计、计划等。
- 来自于先前类似项目的可重用的测试系统和文档。
- 项目和要执行的测试与先前工作的相似程度。

一些由于团队中的人所产生的影响估算的人为因素：

- 受到鼓舞的和鼓舞他人的经理和技术领导。
- 开明的管理层团队，致力于达到正确的质量级别和进行足够的测试。
- 包括个体贡献者、经理和项目利益相关者在内的所有参与者都具有现实可行的期望。
- 项目团队成员具有恰当的技术、经验和态度，特别是经理和核心成员。
- 项目团队的稳定性，特别是没有人员流失。
- 已确立的正面的项目团队内部关系，包括个体贡献者、经理和项目利益相关者。
- 有能力的、能及时响应的测试环境支持者。
- 整个项目组对测试、发布工程师、系统管理员和其他不那么光鲜但却很重要的角色的认可和感激程度（例如，没有“个人英雄主义”文化）。
- 使用具有熟练技能的合同工和咨询师来弥补差距。
- 诚实、守信、透明和公开的被所有个体贡献者、经理和项目利益相关者共享的议程。

最后，还有某些延迟因素经常会增加工作量并延误进度：

- 过程、项目、技术、组织或测试环境的高复杂性。
- 测试、系统质量或项目本身中有过多的利益相关者。
- 有许多子团队，特别是当这些团队的地理位置各不相同。
- 对人员扩充，培训以适应测试团队和项目团队增长的需求。
- 消化或开发新工具、新技术、测试或项目级别上技术的需要。
- 定制硬件的存在。
- 任何对于作为测试工作一部分的新测试系统的需求，特别是自动化的测试件。
- 任何对开发非常详细、清晰的测试用例的需求，特别是在遇到不熟悉的文档标准时。
- 复杂的组件到达时间，特别是对集成测试和测试开发而言。
- 脆弱的测试数据，例如，时间敏感的测试数据。
- 对高级别的系统质量的需求，例如关键任务和安全关键系统。
- 被测系统庞大而复杂。
- 内容全新或大部分内容是新的，这是指无法从先前的测试项目或合适的基准中获得历史数据来进行估算或测试。

在下一个项目里可能会遇到过程、物质、人为和延迟这4个因素中的一部分或全部。这些因素将会影响到所需的资源和时间。因此，当准备测试估算时，应当考虑如何结合这些因素带来的影响。

哪怕只是遗漏了这些因素中的一项也会把符合实际的估算变得脱离实际。经验经常是最后告诉我们这些因素所造成影响的老师，但是如果您是一名聪明的经理，可以聪明地询问一些关于每个因素是否或如何影响你的项目的问题。

有关偏离测试计划的日志会帮助您从经验中学习提高，因为在很多情况下测试计划中的偏离会帮助我们指出哪些估算假设被证明是无效的。在图 3-24 里重新回顾了前面章节中已经看到过的偏离测试计划的测试日志，这里我将专注于特定的和 TIBCO 许可证有关的测试计划偏离部分。

影响	项目名称	记录	变更描述	影响描述
8/10/1998	集成	9/10/1998	TIBCO 许可证频繁过期。从前是每小时，后来是每天，现在是每周	当这个问题发生时，测试中断并被迫重启。这很可能耗费 20 个小时左右的测试时间以及可能 10 人一小时左右投入。这也会降低对测试结果的信心

图 3-24 估算因素举例

在测试计划和估算里，我们已经假设测试环境是稳定的。这种假设性的测试环境是基于所有基础结构都被正确地配置并且没有发生任何假正面或阻塞测试的事件。当然，这种假设是不准确的。这种不准确的假设会导致不准确的估算和错误的计划。

有趣的是在这个记录被录入测试日志后大约两个月左右，我们发现一个许可证号仍然没有在测试开始前被正确地安装。您可能会说：“Rex，如果您准备了一份预测试检查表，您可能已经发现这个问题了。”对于规模较小、较简单、较稳定的测试环境来说这是对的。而在这个项目里，实际上我们有一个跟踪测试用例、测试环境中的硬件、基础设施和软件以及测试人员之间复杂依赖关系的数据库。即使有了这样一个数据库，也只能具体于所跟踪的部分。作为一名经理，您可以也应当尝试缓解测试相关的项目风险。但是，希望能够缓解所有测试相关的项目风险是很危险的，乐观的幻想会导致不准确的、不可靠的估算。

### 3.4.2 估算技术

可以用自底向上和自顶向下的估算方法。自底向上的估算是基于每个任务。例如，单独对每个任务进行估算，然后汇总这些估算得出项目估算。自顶向下的估算则是基于总体项目。例如，对测试工作量进行估算或者同时对高级别测试活动进行估算。

有多种测试估算技术可供使用：

- 直觉、猜测和经验。
- 工作分解结构，例如甘特图和类似微软 Project 的工具。
- 团队评估会议，非结构化或结构化方法，例如德尔菲法、三点法或宽带法。
- 测试点分析，当具有可用功能点时可以使用这种技术。
- 公司标准和规范，这可能影响或在某些情况下决定测试估算。
- 典型的或被认为是典型的总体项目工作量百分比或员工级别百分比，例如臭名昭著的测试人员—开发人员比例。



- 组织的历史和度量，包括公司内已被确定的测试人员—开发人员比例。
- 工业平均值、度量元和其他预测性模型。

让我们更近距离地看一些较常用或有趣的技术。

最常见的也是我认为最有用的估算技术是构建工作分解结构。工作分解结构是将测试项目分层，首先分解到各个活动，然后分解到任务和子任务级别，直至达到足够的详细级别从而使您可以对每项任务的持续时间和工作量做出聪明准确的估算为止。

作为开始可以使用 ISTQB 基础测试过程作为活动分解参考：

- 计划和控制。我推荐把它们分成两个独立的活动，因为控制是持续进行的。
- 分析和设计。ISTQB 基础测试过程把分析和设计看成单一合并的活动，这也同样适用于估算。
- 实现。也可以把这部分内容包括在分析和设计中，就像我经常做的那样，把它分为“测试开发”或“测试准备”。然而，我经常发现，把测试环境配置任务从其他测试开发任务中分离出来是有帮助的，这是因为我的测试环境经常依赖于外部团队。而测试环境配置任务在 ISTQB 基础测试过程中是属于实现阶段的。另外，测试自动化任务在 ISTQB 基础测试过程中也属于实现阶段。如果有大量的测试自动化，那么我倾向于把这部分也作为独立的任务。
- 执行。这是很复杂的部分。这里我经常并不使用细粒度的工作分解结构，实际上我使用工作表来估算执行测试用例所需时间和发现、修复和确认关键缺陷修复所需时间。<sup>11</sup>后面我会举一个具体的例子。
- 结果报告和评估退出准则。可以选择把这部分作为单独的活动，或者也可以把它包括在控制甚至测试执行中，我经常这么做。
- 结束。因为这大都发生在项目结束之后，所以人们可能对这些任务缺乏重视。但是你应该关心这些任务，因为正确的测试结束有助于节省后续项目的许多时间，建立有用的历史度量数据库并且更早地为下个项目制定计划。

现在每个活动都被分解成了分离的任务和子任务。怎么才能知道应该包括哪些任务呢？根据风险分析，您会知道想要测试什么、测多少。因此可以问自己：“哪些测试套件是测试关键风险所需的呢？对于每个测试套件需要哪些任务呢？”对于每个任务自问是否需要识别子任务。在达到最佳粒度级别时，任务应当是简短的，需要一到两天至多一个星期左右。

在图 3-25 中，可以看到一个使用历史度量元来估算发现、修复和确认关键缺陷修复所需时间的例子。这是测试执行估算中关键的部分。在创建这个图表的源工作表里，我使用了一些从过去项目里得到的历史数据以及缺陷发现和修复率的简单模型。因为该模型被参数化，所以它对应于所分配的测试人员和开发人员数量。类似这样的模型依赖于从历史项目中得来的历史数据、公式和常量。因此你有的数据越多，你的模型就会更精确，那么估算的精确性也就越高。

---

11 我的书“Critical Testing Processes”中阐述了这种方法的实例。

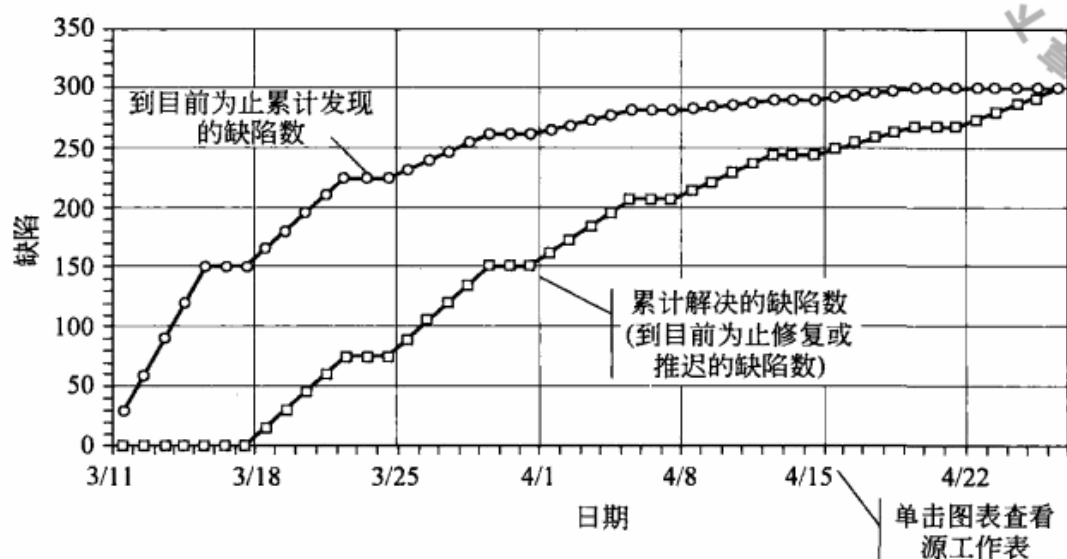


图 3-25 缺陷发现/修复/确认趋势预测

### 3.4.3 使用工业平均值

工业平均值在估算中也很有用。在表 3-15 和表 3-16 里，可以看到 Capers Jones 研究得出的数字。<sup>12</sup>Jones 从几百个客户的几千个项目中收集数据得出他的这些研究成果，从而使得他的数据相当有帮助。

表 3-15 任务、活动和规模估算规则

项目的人一月	(FP/150).FP <sup>0.4</sup>
开发测试用例的工作量	1 人 小时
所需测试用例的数量	FP <sup>1.2</sup>
预计缺陷数量	FP <sup>1.25</sup>
U.S. 每功能点平均缺陷	5.00
缺陷来源率（需求：设计：编码：文档：回归）	3:4:5:2:1
正式设计审查的缺陷移除有效性（DRE）	65%
正式代码审查的 DRE	60%
设计良好的测试套件的 DRE	30%
全项目的 U.S.平均 DRE	85%

Jones 在他的一些数据里用到了功能点。功能点是采用严谨而明确的方式根据应用程序的需求来计算其规模大小。根据 Jones 的研究，也可以把代码行转换为功能点或者相反。可供参考的是一个功能点大约 125 行 C 代码或 55 行 C++代码。

让我们在表 3-15 中看一下一些特别有趣的度量元。例如，Jones 发现平均每个功能点有 5 个缺陷。这相当于每 25 行 C 代码有一个缺陷或每 11 行 C++代码有一个缺陷。

<sup>12</sup> 这部分材料可以在 Capers Jones 的书“Estimating Software Costs”中找到。



缺陷源来自哪里呢？Jones 发现通常来讲 20%的缺陷来自于需求，25%的缺陷来自于设计，33%的缺陷来自于编码，15%的缺陷来自于文档，而 7%的缺陷来自于劣质的修复（回归）。

再让我们看一下 DRE 数字。DRE 或缺陷移除有效性是系统存在缺陷（在某些质量控制活动发生时）和被某个活动移除的缺陷的比例。例如，Jones 发现评审时正式设计审查平均移除了设计中 65%的缺陷。

表 3-16 包括了编码、测试和其他活动的工作量比例。有一点要注意的是这里的测试数字包括了修复缺陷所需的开发时间。

表 3-16 项目工作量比例（测试包括缺陷修复）

编码：测试：其他：（10 FP）	2:1:1
编码：测试：其他：（100 FP）	4:3:3
编码：测试：其他：（1 KFP）	3:3:4
编码：测试：其他：（10 KFP）	2:3:5
编码：测试：其他：（100 KFP）	1:2:3
编码：测试：其他：（最终用户）	6:3:1
编码：测试：其他：（MIS）	1:1:2
编码：测试：其他：（系统）	2:3:5
编码：测试：其他：（商务）	2:3:3
编码：测试：其他：（军事）	1:1:3

### 3.4.4 测试点分析

测试点分析是基于功能点分析对包括系统测试和验收测试在内的高级别测试的工作量进行估算的一种测试估算方法。它也根据 ISO 9126 特征分析来决定对哪些内容进行测试。它是欧洲 Sogeti 咨询公司的 T-MAP 测试模型的一部分。<sup>13</sup>

项目的测试点数量主要依据 3 个要素进行计算：

- 项目的大小，这可以使用功能点根据复杂度、接口和合规性调整后获得。
- 测试策略，特别是关于哪些质量特性或风险需要测试以及需要测试到什么程度。基本的测试策略是假设使用分析式的测试策略。
- 参与者的生产力，这是由测试团队的技术决定，并受到项目、过程、技术和组织因素的影响。

测试点分析模型是一个有趣的模型。即使你无法直接应用它，但是它还是可以被用来帮助你思考如何用历史数据获得自己的测试估算模型。让我们看一下 TPA 过程的步骤。在图 3-26 里可以看到测试点的计算方法。让我们从左上角开始顺时针解读图 3-26。

<sup>13</sup> Martin Pol, et al 的书“Software Testing”中讨论过测试点分析。

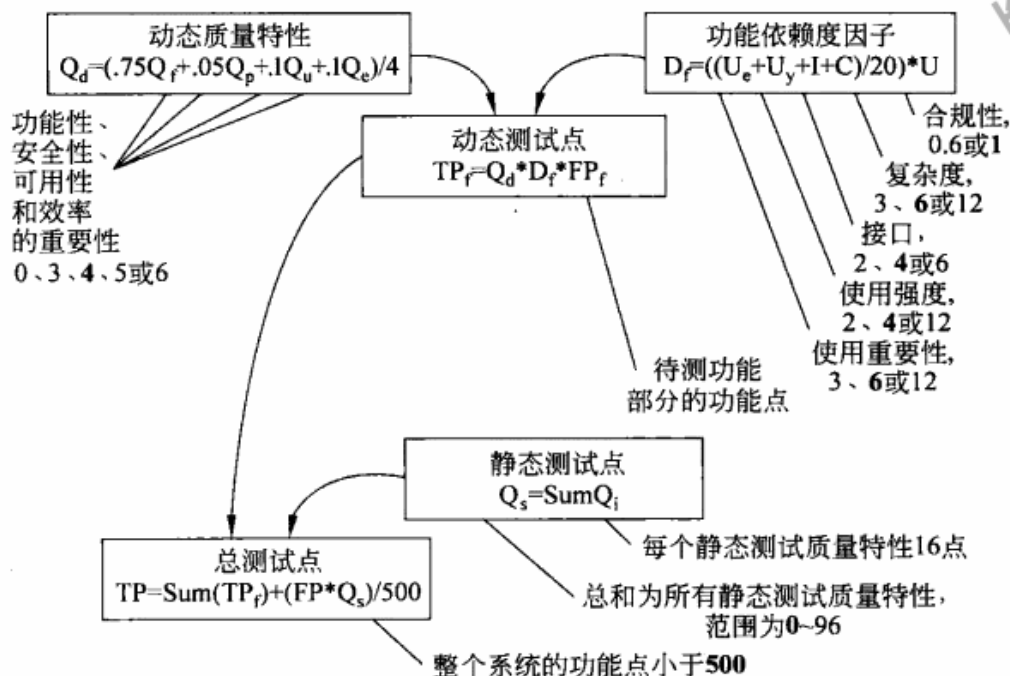


图 3-26 计算测试点

首先, 我们将要为系统的每个功能部分计算出一个数字来代表动态质量特性工作量, 例如动态测试所测试的那些质量特性。动态测试是我们运行系统来进行的测试, 与此相反的是无需运行系统的静态测试。考虑 4 种特性: 功能、可用性、安全性和效率。根据每个质量特性的动态测试重要性, 选择 0、3、4、5 或 6 中的一个数字。越小的数字代表重要性越低, 越大的则代表重要性越高, 其默认值是 4, 在图中被粗体显示。

下一步看右上角, 我们根据各种决定功能模块测试复杂度的参数, 对系统中每个功能部分计算功能依赖度加权因子。对于使用重要性、使用强度、接口和复杂度 4 个参数分别从 3 个数字中选择一个代表数字, 默认值仍然在图中被粗体显示。数字越小意味着测试越简单或越不关键, 数字越大则意味着测试越难或越重要。对于合规性, 你从两个数字中选择一个, 数字越小意味着越难进行测试。

再来到中间靠上的部分, 接下来我们计算系统的这个功能部分的动态测试点。我们用动态质量加权因子乘以功能依赖度加权因子再乘以这个功能部分的功能点数字。重复这个步骤来计算系统中所有功能部分的动态测试点。如果我们面对的是一个复杂的系统, 那么这可能需要很长时间。

现在来到中间偏下的部分, 我们开始计算静态测试点, 这是我们为整个系统进行静态测试所覆盖的测试点。静态测试覆盖的 6 个质量特性来自 ISO 9126, 每个质量特性对应 16 个静态测试点。例如, 如果你会对系统的功能需求进行评审, 那么增加 16 个静态测试点; 如果你会运用静态分析来寻找代码可维护性问题, 那么增加 16 个静态测试点等。静态测试点的范围可以是 0~96。

最后来到左下角的部分, 在这里我们计算出总的测试点。我们对所有功能部分的所有动态测试点进行加总, 然后我们把静态测试点乘以整个系统的功能点再除以 500。

在图 3-27 中可以看到如何使用测试点来计算测试小时数。让我们仍然从图的左上角



开始顺时针解读。

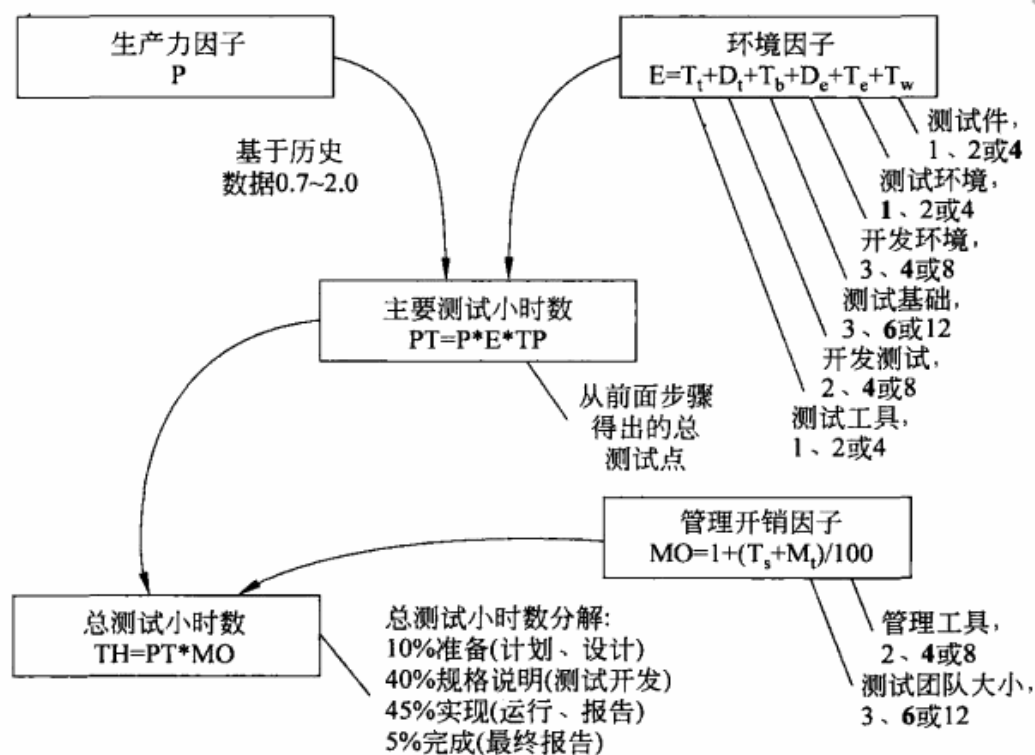


图 3-27 计算测试小时数

首先我们有一个生产力因子，范围为 0.7~2.0。数字越小意味着生产力越高，数字越大则生产力越低。这个因子通过历史数据计算得到。在组织中第一次使用测试点分析时，你需要计算测试点并且将它与前两三个项目的测试小时数进行比较来校准这个生产力因子。

接下来来到右上角，计算将会使测试小时数增加或减少的环境加权因子。我们用测试工具、开发测试、测试基础、测试环境、开发环境和测试件这些隐含属性来评定环境加权因子。数字越小意味着越好，越有助于高效率的测试。数字越大则意味着越差，越可能造成低效率的测试。在这里缺省值也在图中用粗体表示。

在图的中间部分，我们通过测试点乘以生产力因子再乘以环境加权因子计算得到主要测试小时数。

在右下角，我们对管理开销进行计算。这里我们会考虑测试团队的大小和管理工具。数字越大意味着开销越多以及因此带来的更多的测试小时数，而数字越小则意味着管理开销越少。

最后，在左下角我们通过将主要测试小时数和管理开销相乘得出总测试小时数。然后用图底部中间的分解百分比将总测试小时数分布到测试过程中去，这里说的测试过程指的是 T-MAP 过程而非 ISTQB 过程。

这毫无疑问是一个复杂的模型。而且你可能会争论说一些重要的因素，例如外包和离岸开发没有被包含在内。虽然这个模型并不完美，但是这给了我们一些关于如何构建一个复杂的估算模型并且最终根据组织实际情况进行调整的思路。如果你从过去的项目中得到了足够的历史数据，那么你很可能会构建并调校出一套非常准确的测试估算模型。

### 3.4.5 协商和减少测试范围

即使你并非在销售你的服务或是投标，协商通常仍然会是测试估算的一部分内容。你必须让项目经理、项目赞助方、其他项目利益相关者以及任何需要参与批准进度和预算的人确信你所要求的资金、时间和人力是一个好的投资选择。

在本章的后续部分我们将会谈到测试的商业价值，这可以帮助我们来推销我们的估算。但是你仍然经常会遇到必须进行协商的情况。协商可能会导致估算的调整，通常是向下调整。在进行协商时，你应当和项目利益相关者一起尝试达到进度、预算、质量和功能的平衡。

当你尝试达到最佳可能估算时，你可以使用风险分析来降低测试范围。一种方法是削减测试覆盖范围。你可以识别低风险的部分并且在这些部分减少或降低测试覆盖率，这就是“保护深度，减少广度”的方法。

另一种方法是在保持一定覆盖率的情况下降低测试深度。你可以识别最高风险部分中风险最低的部分然后平衡测试深度，这就是“保护广度，减少深度”的方法。

还有一些其他的方法，你可以推迟原计划在项目中开展的自动化或工具开发。当然，这种方法也有它的风险。如果你总是选择协商削减测试过程改进，例如测试自动化，那么你将如何改进测试呢？另外一种可选的方法是识别出有效的部分进行外包，例如兼容性或性能测试。这样能够降低成本、加速进度或两者兼得。

有关预算和进度的协商可能会让你有挫败感，尝试专注于在你最终得到的限制条件内进行尽可能多的测试，有创造性地想出能够工作的方法。同时，确保你自己已经理解了协商的内容。在某些情况下，如果进度可以提前那么会有更多的可用资金，而有时缺乏资金但可以接受一定程度的进度延误。如果两者都是不可协商的，那么可以询问管理层他们是否真的认为进度和预算都要比质量更为重要。如果答案是肯定的，那么或许他们可以减少项目的总体范围？

如果协商的最终结果是减少测试范围，那么应当确保获得利益相关者的批准。同时，参与质量风险分析的人应当被告知风险分析结果和最终将要测试的内容之间的区别。

这也包括耐心，你需要准备反复协商直到完成为止，在这个过程中越少运用到你的政治资本越好。用合适的方式进行协商，让每个人都高兴或至少舒服地感受到现实、准确的估算将会使测试对项目做出重大的贡献。

### 3.4.6 测试估算练习

重新回顾在第1章里对第二个“软件生命周期中的测试”练习所做出的解答。用解答中所列出的活动，对测试工作进行估算。你也应当考虑到你的质量风险分析和测试计划。



向下分解到主要任务（3~10天）。

如果在教室里学习，请将大家分成3~5个小组。如果您是单独学习，需要独自完成这个练习。如果在教室里学习，当每一个小组完成了他们的分析后，对分析结果进行讨论。

我建议用90分钟来完成这个练习，包括讨论的时间。

### 3.4.7 测试估算练习参考答案

我通过两个步骤来解答这个练习。首先，回顾第1章第一部分练习的方案并为每个任务按照人天添加工作量估算。

#### 1. 初始项目计划阶段

##### 1.1 测试计划和控制

1.1.1 生成系统和系统集成测试计划（5天）

1.1.2 参加总体项目计划（5天）

##### 1.2 测试分析和设计

1.2.1 执行质量风险分析（5天）

1.2.2 列出每个迭代中所需测试套件大纲（1天）

#### 2. 迭代一（具有最高优先级的功能）

##### 2.1 测试计划和控制

2.1.1 根据迭代一的需要调整计划（1天）

2.1.2 在迭代一中指导测试工作（持续）

##### 2.2 测试分析和设计

2.2.1 调整质量风险分析（1天）

2.2.2 为迭代一设计测试套件和测试用例（30天）

##### 2.3 测试实现和执行

2.3.1 为迭代一实现测试套件（30天）

2.3.2 为迭代一执行测试套件（6个星期）

##### 2.4 评估退出准则和报告

2.4.1 根据测试计划中的退出准则检查测试结果（持续）

2.4.2 向项目管理团队报告测试结果（持续）

#### 3. 迭代二（具有高优先级的功能）

##### 3.1 测试计划和控制

3.1.1 根据迭代二的需要调整计划（1天）

3.1.2 在迭代二中指导测试工作（持续）

##### 3.2 测试分析和设计

3.2.1 调整质量风险分析（1天）

3.2.2 为迭代二设计测试套件和测试用例（30天）

- 3.3 测试实现和执行
  - 3.3.1 为迭代二实现测试套件 (30 天)
  - 3.3.2 为迭代二执行测试套件 (6 个星期)
- 3.4 评估退出准则和报告
  - 3.4.1 根据测试计划中的退出准则检查测试结果 (持续)
  - 3.4.2 向项目管理团队报告测试结果 (持续)
- 4. 迭代三 (具有中等优先级的功能)
  - 4.1 测试计划和控制
    - 4.1.1 根据迭代三的需要调整计划 (1 天)
    - 4.1.2 在迭代三中指导测试工作 (持续)
  - 4.2 测试分析和设计
    - 4.2.1 调整质量风险分析 (1 天)
    - 4.2.2 为迭代三设计测试套件和测试用例 (30 天)
  - 4.3 测试实现和执行
    - 4.3.1 为迭代三实现测试套件 (30 天)
    - 4.3.2 为迭代三执行测试套件 (6 个星期)
  - 4.4 评估退出准则和报告
    - 4.4.1 根据测试计划中的退出准则检查测试结果 (持续)
    - 4.4.2 向项目管理团队报告测试结果 (持续)
- 5. 迭代四 (具有低优先级的功能)
  - 5.1 测试计划和控制
    - 5.1.1 根据迭代四的需要调整计划 (1 天)
    - 5.1.2 在迭代四中指导测试工作 (持续)
  - 5.2 测试分析和设计
    - 5.2.1 调整质量风险分析 (1 天)
    - 5.2.2 为迭代四设计测试套件和测试用例 (30 天)
  - 5.3 测试实现和执行
    - 5.3.1 为迭代四实现测试套件 (30 天)
    - 5.3.2 为迭代四执行测试套件 (6 个星期)
  - 5.4 评估退出准则和报告
    - 5.4.1 根据测试计划中的退出准则检查测试结果 (持续)
    - 5.4.2 向项目管理团队报告测试结果 (持续)
- 6. 迭代五 (具有最低优先级的功能)
  - 6.1 测试计划和控制
    - 6.1.1 根据迭代五的需要调整计划 (1 天)
    - 6.1.2 在迭代五中指导测试工作 (持续)



- 6.2 测试分析和设计
  - 6.2.1 调整质量风险分析（1 天）
  - 6.2.2 为迭代五设计测试套件和测试用例（30 天）
- 6.3 测试实现和执行
  - 6.3.1 为迭代五实现测试套件（30 天）
  - 6.3.2 为迭代五执行测试套件（6 个星期）
- 6.4 评估退出准则和报告
  - 6.4.1 根据测试计划中的退出准则检查测试结果（持续）
  - 6.4.2 向项目管理团队报告测试结果（持续）
- 7. 项目结束阶段
  - 7.1 测试计划和控制
    - 7.1.1 将与计划不一致的部分文档化（5 天）
    - 7.1.2 参加项目总结回顾（1 天）
  - 7.2 测试结束活动
    - 7.2.1 最终确定需要归档或移交的测试件（20 天）
    - 7.2.2 将测试环境配置文档化（10 天）

有关这些估算的注解：

- 我假设团队是具有技术能力的，能够工作于高级别的逻辑测试用例。
- 我假设每个迭代的内容规模都大致相同。每个迭代包括 2 个人月的测试设计和实现工作，以及共 3 轮、每轮周期为两星期的测试执行（总共 6 个星期的测试执行时间）。

其次，我用微软的 Project 记录整个进度，在各个迭代和主要活动之间建立恰当的依赖关系。如果在课程中没有这样的工具，那么你可能需要在白板上用即时贴来进行工作结构分解。顺便说一句，我随意选择了 1 月 1 日作为项目的开始日期。

有关工作分解结构的注解：

- 我假设测试团队中有一个测试经理，两个测试组长和 6 个测试分析师。
- 因为我没有足够的有关开发工作量或我将会发现的缺陷数量的信息，所以这只是初步的估算。
- 我需要使得以上估算、项目计划和开发计划保持一致。
- 如果我们假设会在测试执行阶段中每周收到一次测试发布，那么我应该尝试将后续每个迭代的分析设计和先前迭代的分析设计相重叠，从而使得执行总是能够从星期一开始。

我用微软 Project 创建的甘特图如图 3-28、图 3-29 和图 3-30 所示。

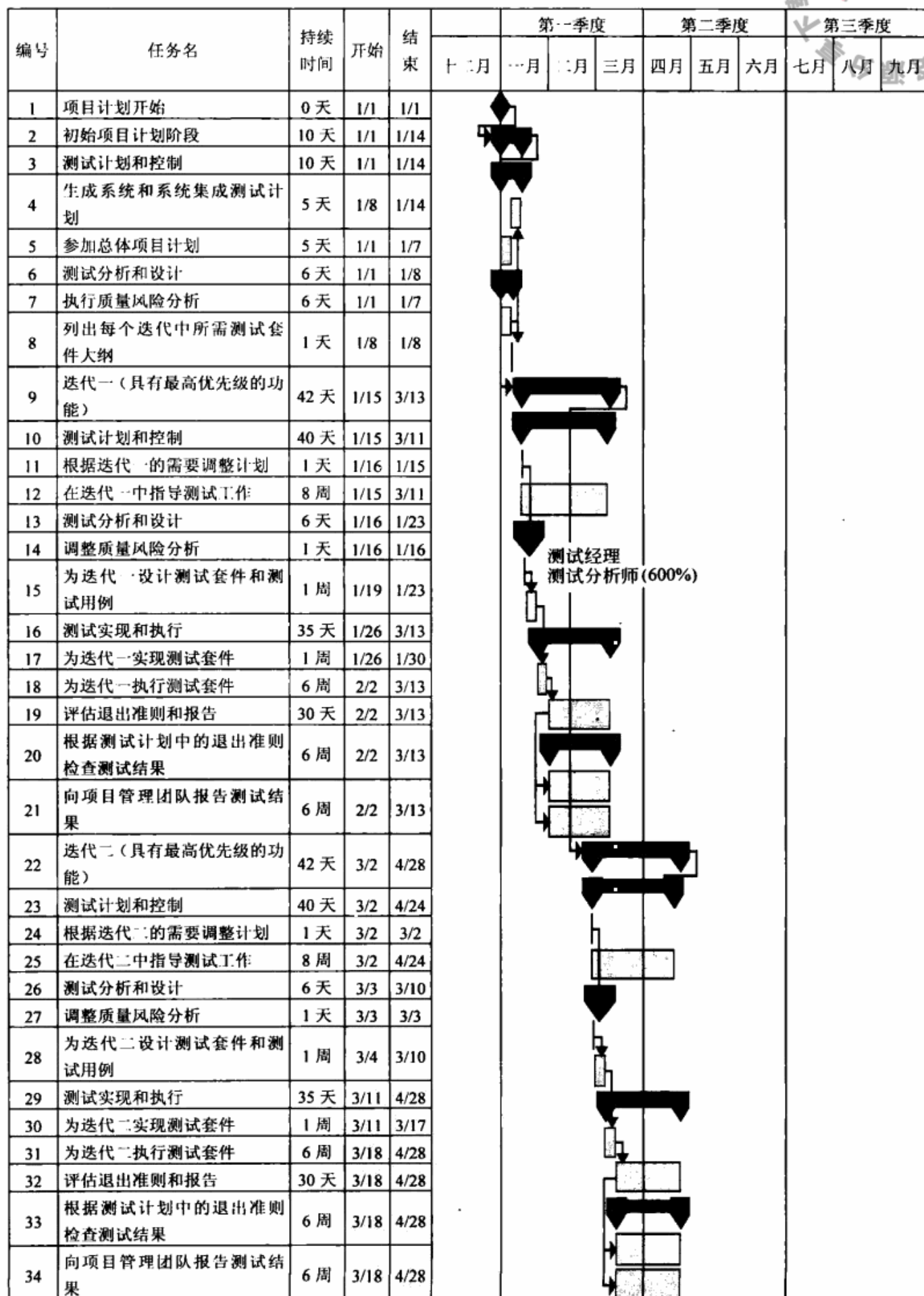


图 3-28 计划阶段、迭代一和迭代二的甘特图



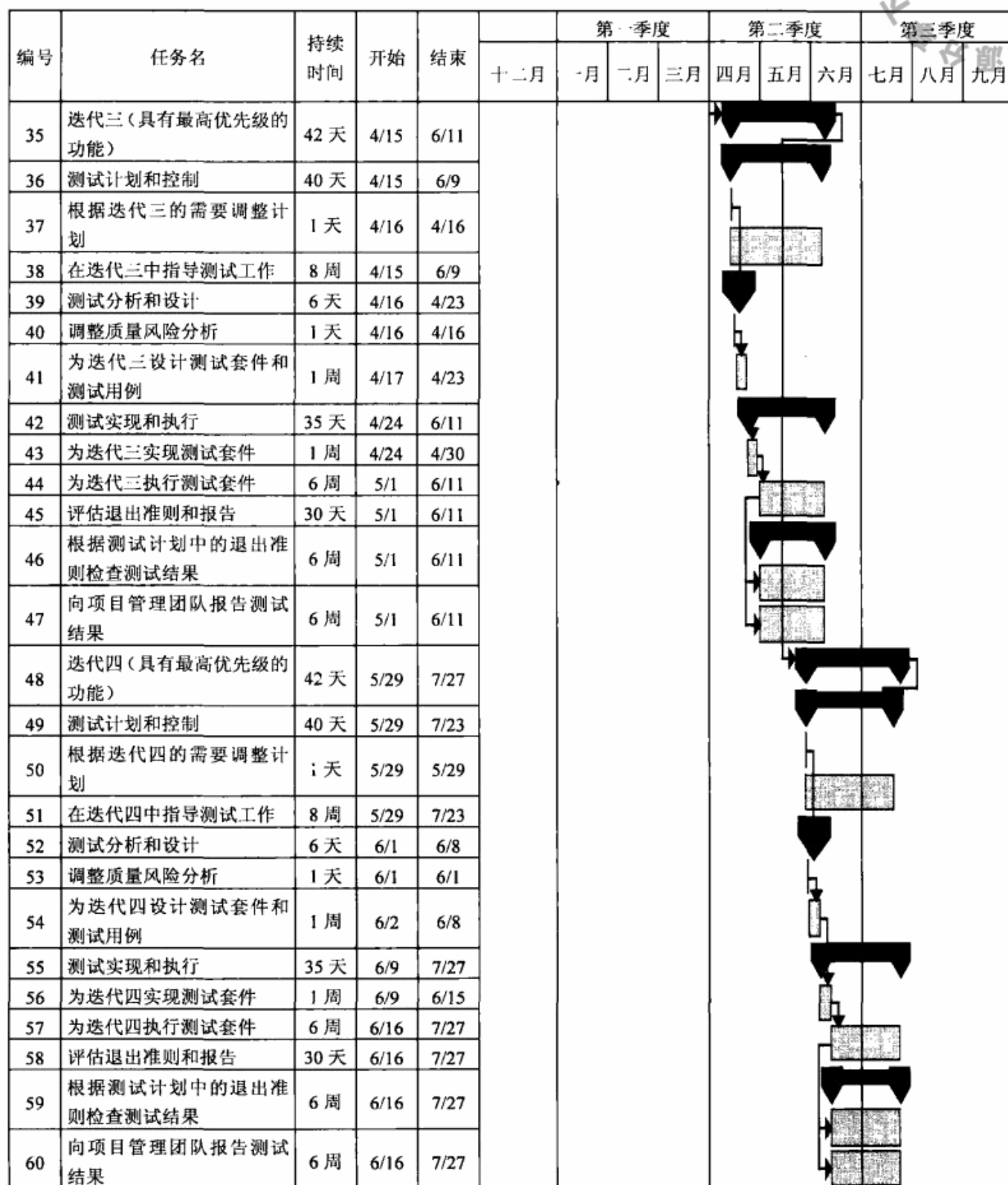


图 3-29 迭代三和迭代四的甘特图

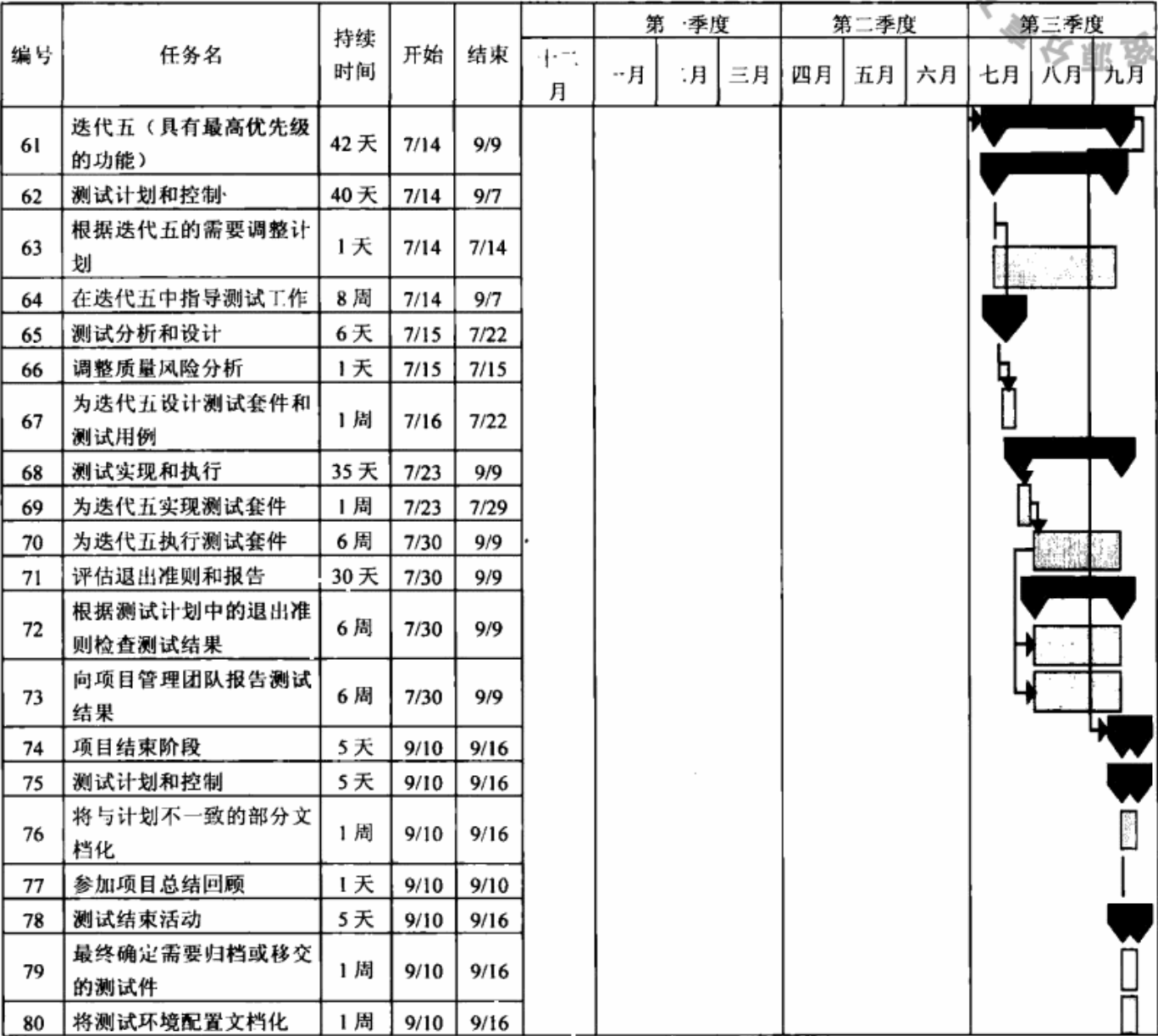


图 3-30 迭代五和项目后阶段的甘特图

### 3.5 测试计划安排

📖学习目标:

(K2) 描述尽早制定测试计划和进行测试计划迭代的益处用实例来支持你的解释。

美国将军和总统德怀特·艾森豪威尔曾经说过：“在战斗准备中我经常发现计划毫无用处，但是进行计划是必不可少的。”项目不像战斗这么残酷，至少通常没有人会在软件项目中死去，但是项目和战斗有许多相同的特性。其中之一是两者都需要尝试尽早地发现其中重要的特征性，从而能够针对这些特征采取有用的行动并且尽早地识别出关键风险加以缓解，或者至少在这些关键风险成为负面事件前采用应急性计划。

提前进行计划会带来许多益处。其中之一便是更早地发现和管理风险，经常会使风



险的发生可能性和潜在影响降低。另一个益处是提供一个和其他项目参与者进行仔细和及时交流协作的机会,从而确保每个人能够各司其职。再有就是发布一个高质量的计划可以为将来的工作提供有帮助的路线图,这些提前进行测试计划的益处对于任何项目都是有效的。

提前进行测试计划还提供了其他的益处。我们经常可以在测试计划阶段发现测试范围之外的项目风险和问题,例如可能在生产环境中造成运行问题的不良假设。及早发现问题使得项目团队可以在这些问题完全成长为生产危机前对这些风险和问题进行管理。

在测试计划中,我们经常可以通过质量风险分析在测试执行前发现质量风险和问题,例如在产品需求中缺少性能和可靠性信息。我们可以在质量风险文档中提供这些信息并把它们应用于测试设计、实现和执行过程。我们也可以向项目团队提供这些信息从而使他们可以采用适当的方法来解决这些问题。

在测试计划阶段,我们经常可以在项目计划和其他项目工作产品中识别到一些问题,例如项目团队缺少诸如可靠性、可用性、本地化或性能等特定的技能。我们可以向项目团队提供这些信息从而使他们可以采取的措施来解决这些问题。

在测试计划阶段,我们还可以找到机会调整测试人员、预算、工作量或持续时间,根据特征、进度和预算的限制更好地平衡质量优先级。

在早期测试级别的测试计划阶段,我们可以识别到例如服务器、数据库或核心软件元素等关键组件。这些信息使得我们可以进行基于风险的集成,计划对这些关键组件更早地进行单独测试和集成测试,尽早地发现重要的集成问题。

通常来说,测试活动位于其他项目活动的下游,而帮助台或技术支持、技术发布、生产运行和项目赞助方则位于测试活动的下游。测试不会,也不应当或不可能独自存在。孤立的测试本身是没有价值的,测试仅有在向其他项目利益相关者进行某些交付时才有其价值。测试本身是无法单独产生作用的,因为它包括对其他人交付和构建的测试项进行评估,而测试环境也经常会被其他人进行配置和支持。

因此,测试计划中的一个重要元素就是识别出测试和项目的接触点及传递。以下所列是一些接触点和传递的例子:

- 在测试实现中尽早交付非正式版本,对测试用例、测试规程、测试环境和测试工具进行测试。
- 参考 IEEE 中有关测试项传递部分,用定期的、有组织的、可靠的、高效的和最低分散程度的方式来协调正式的测试版本发布。
- 确保测试发布的修订号是可靠、递增和易查明的,从而可以收集和报告有意义的测试结果。
- 当测试环境的复杂度、规模或危险程度超出测试团队的能力范围时,对测试环境提供可靠、及时、有效的支持。

所有这些任务都包括了相互达成一致的过程,因此你必须和其他项目参与者一起协作制定计划。

即使你积极地与其他项目参与者进行接洽尝试获得测试计划所需的信息,如果你等待获得所有的信息,那么你通常会在测试执行开始前一天才完成你的测试计划,甚至更

晚。我曾经遇到过一些项目在项目结束后仍有一些测试计划中的问题未被解决。如果你坚持要获得所有问题的答案后再编写测试计划,那么这就意味着无法提前进行测试计划。

我的方法是尽早地编写测试计划并发布草稿版本,然后与关键的测试参与者以及利益相关者进行讨论。当获得新的信息后,我会更新草稿版本。如果有我迫切需要的信息或必须要快速解决的问题,我会不断要求和逐级上报直到我获得答案为止。然而这可能会使其他项目利益相关者不高兴,从而迫使我使用政治资本,因此我只会在必要的时候使用这种方法。

我发现在计划过程中运用迭代的、苏格拉底的方法来进行测试计划生成、发布和评审工作是非常有效的。它把测试计划转变为物理手段来提升有关测试的沟通、讨论和合规性,特别是明确项目或测试级别中有关测试的 5W (hows、whos、wheres、whens、whats)。<sup>14</sup>

### 3.5.1 尽早进行测试计划的案例学习

让我们看一个做早期的迭代的测试计划是如何帮助项目在早期识别出许多重要的例子,仍然是用本书前面多次提到过的那个基于因特网的项目,我发布了两个测试计划的草稿版本,每个版本中都提出了一些问题。图 3-31 是和第一版草稿一起发送的说明邮件,这些草稿激励了大家在邮件和会议中讨论和解决这些问题,接下去的几页我会给出几个我在测试计划草稿中列出的确切的问题实例。

各位,  
附件是测试计划草稿第一版,其中包括了硬件(DVT 和 PVT)和软件(集成和系统测试)部分。现在我们遇到了许多问题,希望在明天的评审会议上我们可以解决大部分的问题,或者至少解决必要的问题使我们可以继续向前。  
Ken,接下来我将会在今天下午和明天设计预算和详细的进度。这些都会基于测试计划,所以如果第一次评审有任何问题让我们尽快地一起进行讨论。  
  
致敬  
Rex

图 3-31 尽早进行测试计划

图 3-32 是我对硬件相关的质量风险提出的问题。我要求 Ed、Qadeer 和 Greg 提供所列质量风险的优先级以及增加一些其他的质量风险。这个问题对于决定硬件相关测试的运行顺序十分关键(因为受到空间限制,许多质量风险没有在图中列出)。

图 3-33 是 3 个关于测试环境的问题。第一个问题是有关产品验证测试所需的装置数量,产品验证测试是硬件相关测试中最高级别的测试,也是宣布硬件测试结束前的最后一个测试级别。我提出是否可以接受用手工制造的工程模型来进行验证,我相当肯定得到的答案会是:“不,我们必须使用生产线装置。”,但我希望能够有人确切地给出这个答案。

<sup>14</sup> 参加过法律学院学习的人都知道苏格拉底的方法是通过提出一些引发思考的问题来帮助人们学习。卡尔顿大学已经在 [serc.carleton.edu/introgeo/socratic/](http://serc.carleton.edu/introgeo/socratic/) 网站上提供了一些很好的解释和例子。



因特网应用程序以存在于个人电脑中的同样类型的质量风险为准。【Greg/Ed/Qadeer: 请帮我对这些项设定优先级并指出任何遗漏的风险。】		
质量风险分类	质量风险项	优 先 级
可靠性	婴儿死亡率	???
	过早失效	
	电池开发“记忆”	
	屏幕退化（像素死亡）	
辐射	超出监管规格说明	???
安全性	尖锐的边缘	???
	电气化部分	
	腕管/RSI	
	幼儿/婴儿/宠物问题	
	不舒服的过热点	
{9 个其他风险分类没有在这里显示……}		

图 3-32 质量风险优先级问题

产品验证测试（PW）入口准则部分：“供应商为 PVT 测试提供 PVT 装置，数量已经在这个计划中明确写明。（参见测试配置和环境）【Greg/Qadeer/Ed: 我们是使用在生产线上大规模生产的 PW 装置还是可以使用工程模型实例进行测试？】”

测试配置和环境部分：“【Greg/Ken: 我们需要定义测试实验室，我还没有和其他人讨论这个问题。我应该和谁谈？它应该被放在哪里？】”

测试配置和环境部分：“【Greg/Qadeer: 根据 Greg 的第一次估算，我们将会有 50 个 DVT【开发验证测试】装置和 100 个 PVT 装置。MTBF【平均失效间隔时间】演示需要大量的装置。同时，如果我们运行 ALT【加速寿命测试】，这些测试将会使装置损坏，因此我们必须把一些测试所需的消耗计算在内。】”

图 3-33 测试环境问题

第二个问题是关于测试实验室的所在地，以及对实验室进行配置和支持的人。

第三个问题和所需的测试装置数量有关。众所周知，平均失效间隔时间测试会占用大量的硬件资源。加速寿命测试不仅会占用硬件资源，而且因为它们会使用资源直到出现失效，它们实际上会使测试资源衰竭。我在这里要求 Greg 和 Qadeer 对我给出的有关这些数字的看法做出确认。

图 3-34 是我有关测试设计的问题。首先，我概要地描述了当前对于服务器端测试所需架构和组件的看法。然后，我向两位资深开发经理/架构师 Jamie 和 Wayne 问了两个问题。我要求就我们如何最终确定服务端测试架构和组件与他们进行讨论并将此电子邮件发给软件供应商，对于将会决定是否批准我的测试预算的决策者 Greg 和 Ken，我申请批准我的含有 COTS 测试工具价格的预算。

【在服务器端，我们可以使用一些 COTS 测试工具。例如，SilkPerformer……可以测试网络站点。同时，【电子邮件软件供应商】应该对电子邮件方面的测试提供测试工具或直接的测试帮助，这会留下更新和状态接口。我们应当能够用另一台运行在 Linux 上的 32 位的个人电脑通过 LAN 连接到服务器进行测试。该测试工具探测到其软件或数据级别低于当前版本时，能够触发从服务器的下载，它从服务器获得新的发布但将数据发送到/dev/null。

关键的问题是如何模拟 500 000 用户级的负载。我们也许可以生成一些其他的服务器端通信负载工具来加载特定的子系统让它“看上去像”500 000 登录用户的情况，例如数据库或 LAN 基础架构。但是我们必须非常谨慎地设计一些性能模型，想清楚怎么做。Wayne/Jamie，我们是否能够让【电子邮件软件提供商】也加入进来，对他们的这部分内容加以理解，然后一起对负载生成器进行高级别的设计？Greg/Ken，是否可以批准我所建议的网络站点 COTS 方案预算？】

图 3-34 测试设计问题

## 3.6 测试进程监控

### 学习目标:

(K2) 比较测试进程控制的不同流程。

(K2) 从概念上至少给出 5 个不同的例子，解释测试进程中发现的问题如何影响具体的测试过程。

(K4) 利用在监控活动和测量中所观察到的测试进程相关的问题，制订应对计划改进当前的测试过程，并提供改进建议。

(K4) 分析测试结果和确定测试进度，并记录到监控报告和最终的测试总结报告，从 4 个维度生成总结报告的内容。

测试进程监控和报告具体细节每个组织都不一样，我已经观察到了测试进程监控和报告的 5 个主要方面：

- 质量风险。
- 缺陷、事件。
- 测试用例或测试规程。
- 测试基础、代码或其他感兴趣部分的覆盖率。
- 信心。

前面 4 项——风险、缺陷、测试和覆盖率是可度量和可量化的，甚至有时候有太多方法可以进行度量。我说“太多种方法”的意思是测试经理有时候会关注于一个或两个方面，最常见的就是缺陷和测试这两个方面，然后生成许多不同的视图、分析和图表。而这样做的有效性会低于考虑所有 5 个方面的整体分析方法。

当考虑到所有 5 个方面时，测试经理应该尝试对每个方面进行度量并且报告结果。对于信心，我们经常对其进行主观的判断或使用一些替代性的度量，最常见的是一些需求覆盖率、缺陷发现率和缺陷积压规模的组合度量。



使测试进程监控的这些主要方面和测试完整性评估保持一致是很重要的。我们应该使测试进程度量和我们定义的退出准则保持一致，反之亦然。

让我们用一些样例图表来看一下4个可以直接度量的方面。我将会使用一些实际的学习案例作为例子，大部分例子都基于一个假定的单独电子商务应用程序测试。

### 3.6.1 产品风险度量

如果采取基于风险的分析式测试策略，那么可以根据风险对测试进行度量。从负面的角度，可以根据剩余风险数量和这些风险的细节，例如类型和级别来进行度量。从正面的角度，可以根据已缓解风险的数量和这些风险的细节，例如类型和级别来进行度量。可以用 Excel 或类似工具根据测试和缺陷的覆盖率数据，手工把这些内容简单地集中在一起。现在一些测试管理工具也已经开始直接包含这样的功能。

图3-35在前面基于风险测试的章节中已经出现过。让我们回顾一下它是如何工作的。中等灰色区域表示所有相关测试已经被运行并且没有发现必须修复缺陷的风险。浅灰色区域表示至少有一项测试失败并且至少有一个必须修复缺陷的风险。暗灰色区域表示其他的未知有必须修复的缺陷，但有部分测试尚未执行的风险。

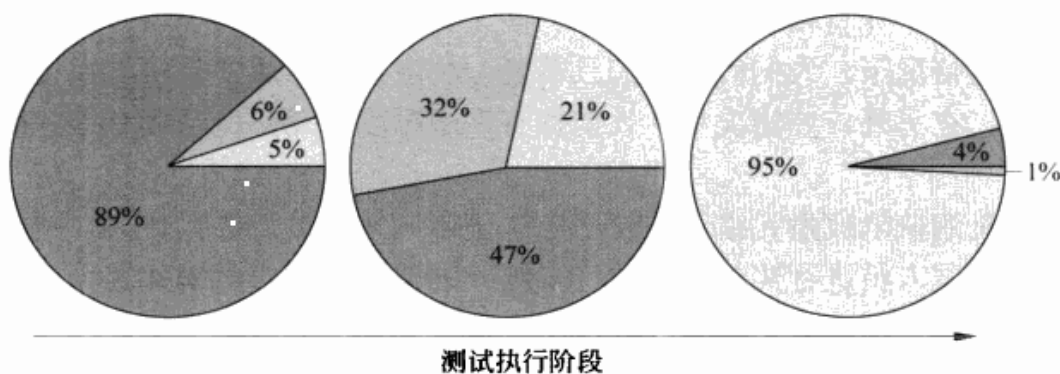


图 3-35 产品风险度量

底下的时间轴是表示当我们开始测试时，饼图完全是暗灰色的。随着测试执行的展开，我们可以看到浅灰色部分逐渐开始占据暗灰色区域，而中等灰色区域也开始占据浅灰色部分。当我们完成测试执行时，大部分暗灰色和浅灰色区域将不复存在。

对于中等灰色的区域，在任何时间我们都可以生成一个详细的列表，包括测试过的风险，已知的通过测试的风险以及未发现任何必须修复缺陷的风险。对于浅灰色部分，我们可以生成一个详细的已知有测试失败并且有必须修复缺陷存在的风险列表。对于列表中的每个风险我们可以列出相关的测试和缺陷。对于暗灰色部分，我们可以生成一个详细的测试未执行的风险列表。这可以帮助我们准确地了解风险，降低成果和剩余的风险级别。

这种类型的度量元并不常见，但这就是高级经理、执行官和那些测试以外的人希望从测试中获得的战略性的关键信息类型。

### 3.6.2 缺陷度量元

当我们谈论缺陷度量元时，以下这些是最常见也是经常被滥用的度量元。常见的缺

缺陷度量元包括累积报告缺陷数量和累积解决缺陷数量。当项目临近结束时,累积报告缺陷数量增长趋势变缓(标志着低缺陷发现率或零缺陷发现率),而累积解决缺陷数量应当趋近于累积报告缺陷数且最终与其交汇。

在安全关键和任务关键系统中,人们会对失效更感兴趣而对缺陷的关心程度较低。所以我们经常看到有关平均失效间隔时间或其他失效到达率的度量元。

通过分类对缺陷进行分析或分解也是很常见的,特别是在尝试理解缺陷为什么会发生的时候。我们可以根据测试项、组件或主要子系统进行分类,这可以帮助我们找出缺陷集中区域。我们可以根据根本原因或缺陷来源进行分类,这可以帮助我们降低缺陷引入率。我们可以根据测试发布进行分类,我们希望这会随着时间的推移显出向下的趋势,但这其中也可能包括某些出乎意料的尖状突出部分。我们可以根据缺陷被引入、发现和消除的阶段来进行分类,这可以帮助我们找出降低缺陷成本和进度影响的机会。最后,我们也可以根据类别本身进行分类。

另一种常见的缺陷度量元是关闭周期。这种图表表现了从缺陷报告到缺陷修复间隔时间的趋势。

我发现这些缺陷度量元以及对趋势和分析的图表化方法对于理解项目、产品以及过程的行为和问题是十分有帮助的。但是,有些人将这些度量元滥用为一维的度量元。我的意思是在某些情况下,人们假设他们只需要一种度量元,通常是累积报告缺陷数和累积解决缺陷数。在另外一些情况下,人们拥有许多种度量元,但是所有这些度量元都基于缺陷。

另一种常见的缺陷度量元错误的使用方法是用它来对员工进行评估。例如,根据缺陷发现率来对测试人员进行奖惩会导致许多细碎的缺陷被报告出来并且会让开发人员非常愤怒。根据缺陷关闭周期来惩罚开发人员会造成惩罚了最好的开发人员(这些开发人员经常会被分配到最难的缺陷),而鼓励了草率和拙劣设计的“绷带式”的缺陷修复。

让我们看一些基于缺陷的图表例子。图 3-36 是一种数据丰富的累积报告/累积解决图表的变体。这个图表主要有 5 个数据集,其中两个基于右边轴计量,另外 3 个基于左边轴计量。

第一个数据集被显示为上方的实线,它代表了累积报告的缺陷数量,该图表基于右边轴计量。第二个数据集被显示为下方的实线,它代表了累积解决的缺陷数量,它也是基于右边轴计量的。因为该图表的目标是为了指出缺陷的状态趋势是否朝着发布准备就绪的方向发展,所以累积解决的缺陷数量包括修复的缺陷数量和延期的缺陷数量。

第三个数据集被显示为灰色的减号,它代表了一个给定日历周中的平均缺陷报告数量,它是基于左边轴进行计量的。第四个数据集被显示为灰色的加号,它代表了一个给定日历周中的平均缺陷解决数量,它也是基于左边轴进行计量的。

第五个数据集被显示为黄色的星号,它代表一个给定日历周中的平均积压数量。积压数量是总缺陷报告数量减去缺陷解决数量。

因为趋势图表现的是时间趋势而非因果关系,所以这个图中包含了注释来解释各种曲线的形状。例如在感恩节假期中累计的缺陷报告曲线是平的,如果没有注释,读者可能会根据自己的理解对过程信息做出错误的解读,在这种情况下通常都会基于产品质量



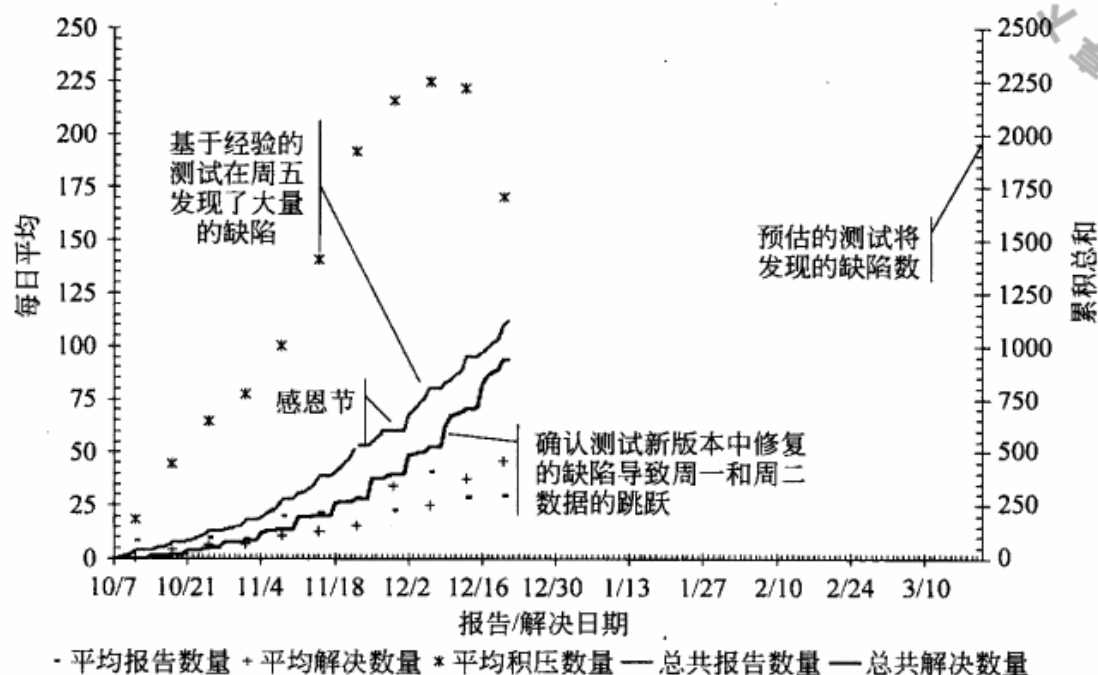


图 3-36 累积报告和解的缺陷

问题的现状做出错误的解读。

图 3-37 是根据缺陷发现的子系统进行的帕雷托分析。我们可以看到大部分缺陷是在这个假设的电子商务应用程序的目录管理、用户接口和签出子系统发现的，这个比例达到了 80%，而剩余子系统缺陷相对要少得多。

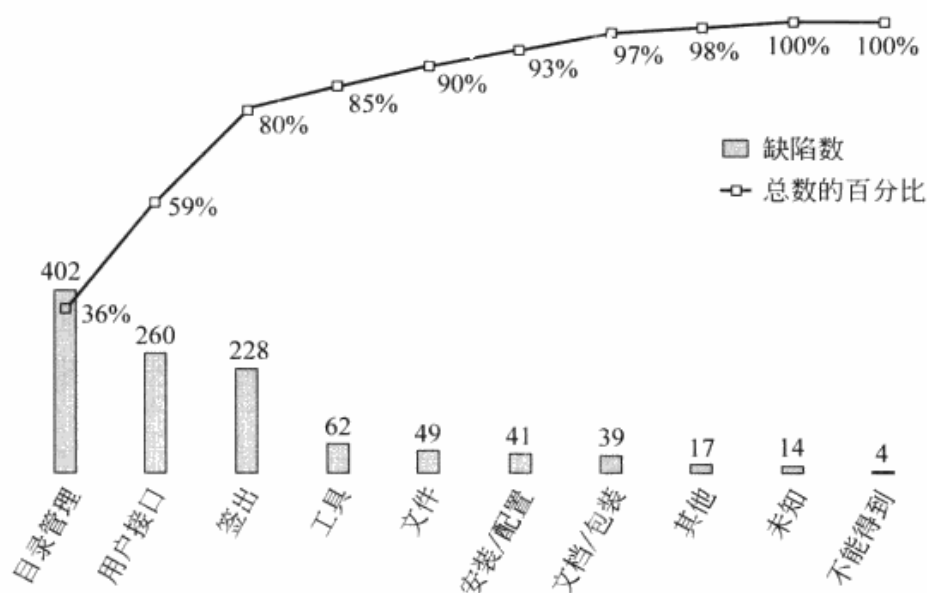


图 3-37 子系统缺陷分解

顺便说一句，帕雷托分析是一种类似上图的柱状图，我们从中可以看到一些根本原因间的联系的发生频率和与该原因相关的结果数量。在上图中，根本原因是指缺陷显露为失效所在的子系统，而与原因相关的结果数量是指根据子系统报告的缺陷数量。柱状图是被排序的，所以最常见的原因被显示在最左边，然后剩下的原因按照发生频率的降序依次向右排列。它包括了上方的累积百分率，它显示了和下方原因相关的结果百分

比以及在下方原因左边的所有原因。帕雷托图是一种非常有用的分析性技术，因为它将其创造者朱兰称为“关键的少数”原因从“细碎的许多”原因中分离出来。我们应当专注于关键的少数，而非细碎的许多内容，这个图可以帮助我们做到这一点。

图 3-38 展现了关闭周期趋势。关闭周期在这个图里用两种方式显示：

- 任何给定日期的滚动关闭周期值是针对所有已解决的缺陷而言，给定日及给定日前的从缺陷报告日到缺陷解决日的平均日历日数。
- 任何给定日期的每日关闭周期值是针对所有已解决的缺陷而言，给定日是从缺陷报告日到缺陷解决日的平均日历日数。

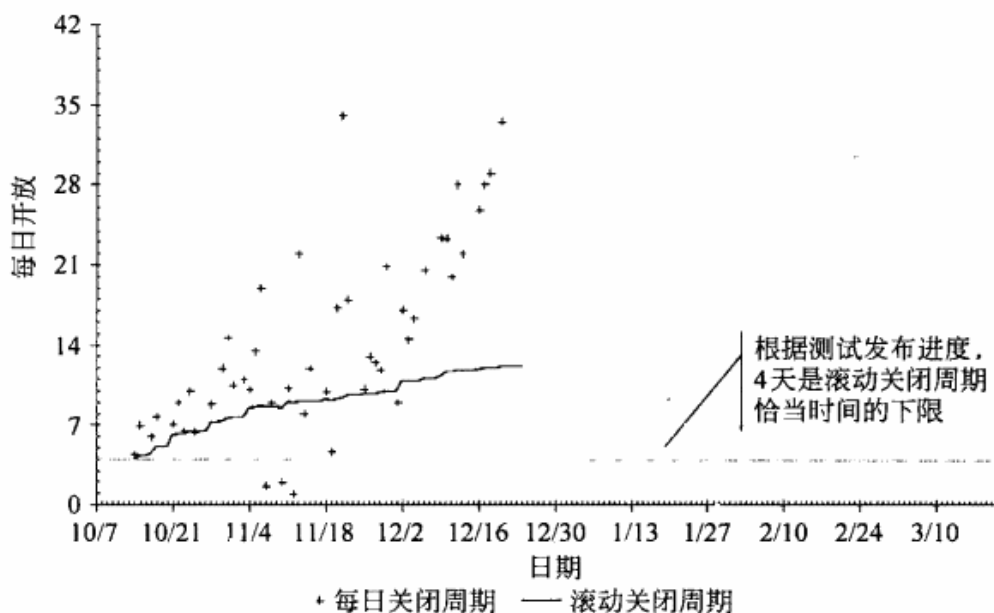


图 3-38 关闭周期趋势

因此，我们会希望滚动关闭周期根据周转时间稳定在某个可以接受的数字。如图所示，我们注意到测试发布过程给出了这个周转时间的下限。

一旦滚动关闭周期稳定在了一个可以接受的数字以后，我们会希望每日关闭周期随机地在滚动关闭周期上下一定范围内跳动。如果每日关闭周期持续地处于滚动关闭周期上方，这意味着项目中发生了某些状况，造成缺陷解决的时间变得越来越长。

### 3.6.3 测试用例（或规程）度量

测试用例或规程度量是另一个十分常用但也经常会被错用的测试度量形式（虽然不同的人会对这些度量类型使用不同的用词，但因为我个人倾向于使用测试用例这个用词而非测试规程或测试脚本，所以在这里我将会用测试用例来进行描述）。一种常见的测试用例度量会对计划的测试总数、编写的测试和运行的测试进行计量和比较。运行的测试有时还会被进一步细分为通过、失败、阻塞和跳过。

另一种常见的测试度量关注回归和确认测试状态。我们也可以通过已报告缺陷的重开次数等缺陷度量来评估确认测试状态，再有一种常见的度量会关注计划测试小时数和实际测试小时数之间的比较。

我们可以用这些度量来理解项目、测试和开发过程。但是，人们经常会错用这些度



量。其中一种错用是过度依赖，这是指依赖于单一的测试度量或单一的缺陷度量来描述整个测试结果，这种方法会完全遗漏关键风险问题和覆盖率。

另一种错用是假装或猜测累积测试用例总数，通过和失败的百分比可以告诉我们产品的质量。问题在于测试用例是一种抽象的概念。有时 40% 的测试用例失败可能并不一定意味着产品质量灾难，但有时 95% 的测试用例通过却会是产品质量灾难，它决定于具体失败的测试。

与缺陷度量一样，另一种典型的度量错用是用它来评估员工。如果我们因为许多测试在运行时失败就说某些开发人员不好，这会造成开发人员对未来所有的测试结果都产生很强的防御性反应。

让我们看一些测试用例度量的例子。在最低的级别上我们需要一些方法来追踪测试用例。虽然许多人已经使用工具来进行测试用例追踪，但确实也有许多的测试经理仍然使用普通的工作表作为追踪方法来管理测试工作量。因为出现在完整的测试追踪工作表里的能力都可以也应当在完整的测试管理工具中出现，所以接下来我将会讨论工作表，而把具体将其映射到任何你可能使用的工具的任务留给你。

图 3-39 显示了我们假设项目的部分测试用例总结工作表。它展现了在许多测试套件中某个测试套件里的测试用例（记住，测试套件是一组测试用例的逻辑组合）。

以下每列为每个测试用例提供了重要的信息：

- 所有者：负责运行测试用例的人。在本例中我们使用姓名字母缩写而非全名。
- 测试号：整数位表示测试套件数字，3 位小数表示测试用例数字。它唯一标识测试用例，无论在缺陷跟踪系统还是其他任何你可能使用的覆盖率跟踪工具里它都是十分有用的。
- 测试套件/用例：这一列首先包括了测试套件名字，然后逐列显示该测试套件下的测试用例的名字。本图只是部分工作表，在完整的工作表里你可以看到，完整的测试套件系列包括每个测试套件相应的测试用例。
- 状态：如果我们已经运行了测试，状态是通过、警告或者失败。通过意味着我们没有发现失效。警告意味着我们确实发现了失效并且我们已经递交了缺陷报告，但是这个失效并没有严重地影响被测试的特征或功能。失败意味着我们发现了失效并且提交了缺陷报告，该失效严重地阻碍了测试的特征或功能。如果我们还没有运行或无法运行的测试，状态应是阻塞。如果我们选择不运行或将不运行的测试，状态应是跳过。如果我们还没有运行测试但是我们仍然计划将会运行这些测试，状态列将会为空。
- 测试环境：为运行测试所用的硬件、软件和基础结构提供识别信息。据推测，在其他追踪工作表或测试计划里的某个地方可能会对该识别信息进行解码，从而决定测试环境中确切包含的内容。
- Def ID（或称缺陷号）：从缺陷跟踪系统对测试中报告的任何缺陷提供缺陷号。
- Def RPN（或称缺陷风险优先级数字）：这里指缺陷风险等级，很可能来自于缺陷跟踪系统，类似于质量风险分析中的风险优先级数字。一种常见的计算方法是用

所有者	测试号	测试 套件/用例	状态	测试环境	缺陷号	缺陷风险优 先级数字	运行者	计划 日期	实际 日期	测试 工作量	计划 工作量	实际 工作量	测试持 续时间	注释
	1.000	功能性												
DLK	1.001	浏览	失败	SW/H2/B1	1832	1	DLK	12/16	12/16	2.0	2.0	3.0	4.0	
JW	1.002	加入购物车	警告	SW/H2/B1			DLK	12/16	12/16	0.5	0.5	1.0	3.0	
BEZ	1.003	从购物车移走	失败	SS/H2/B7	1832	1	BEZ	12/16	12/16	0.5	0.5	2.0	4.0	
DLK	1.004	改变数量	通过	SS/H2/B5			DLK	12/17	12/17	3.0	3.0	3.0	5.0	
BEZ	1.005	登录账号	通过	SW/H1/B6			BEZ	12/16	12/16	2.0	2.0	2.0	4.0	
LJ	1.006	查看订单	失败	SW/H1/B8	1832	1	LJ	12/16	12/16	2.5	2.5	3.0	3.0	
LJ	1.007	提供信用卡	失败	SS/H2/B8	1832	1	LJ	12/17	12/18	4.0	4.0	6.0	6.0	
					1837	6								
					1838	25								
					1840	6								
					1845	3								
					1846	2								
					1847	8								
BEZ	1.008	提供地址	失败	SS/H1/B6	1836	1	BEZ	12/17	12/17	2.5	2.5	6.5	5.0	
					1832	1								
					1841	5								
					1844	15								
BEZ	1.009	确认订单	失败	SE/H2/B6	1836	1	BEZ	12/17	12/18	3.0	3.0	5.0	5.0	
BEZ	1.010	购物车超时	通过	SW/H2/B5			BEZ	12/18	12/19	3.5	3.5	4.0	4.0	
LTW	1.011	收藏管理	失败	SS/H1/B2	1833	2	LTW	12/16	12/16	3.5	3.5	4.0	4.0	
HS	1.012	缺货订单	失败	SS/H2/B3	1833	2	HS	12/16	12/16	3.0	3.0	3.0	3.0	
HS	1.013	删除账号	失败	SE/H2/B3	1833	2	HS	12/17	12/17	3.5	3.5	3.5	3.5	
HS	1.014	选择加入/选择退出	失败	SW/H1/B3	1833	2	HS	12/17	12/17	2.5	2.5	3.0	3.0	
		套件小结						12/18	12/19		36.0	49.0	56.5	

图 3-39 测试用例总结



优先级乘以严重度，其中优先级是指缺陷的商业等级，而严重度则是指系统或技术的等级。在本例中我们将会对优先级和严重度都使用 1~5 的递减标准，例如数字越大意味着风险越低。

- 运行者：最初运行测试的人。
- 计划日期：当测试经理决定测试顺序后在测试执行中定义的测试运行计划日期。
- Act Date（或称实际日期）：测试人员实际运行测试的日期。
- 测试工作量：从开始到结束运行整个测试所需的人—小时数。
- 计划工作量（或称计划的工作量）：该测试计划的人—小时数。如果有额外的探索时间的话，这可能会略大于测试工作量。它也可能比测试工作量略小，这意味着测试人员应该选测试中覆盖的条件。
- 实际工作量：最终花在测试上的人—小时数。这可能会和计划的工作量不一致，特别是如果测试失败和测试人员需要花费大量的时间来分离和报告所发现的问题时。
- 测试持续时间：从测试开始到结束的所需小时数。对于自动化的测试，这会和工作量有很大的不同。

现在您很可能已经认出这是一份逐个测试的测试日志详细报表，而这是对我们前面讨论过的逐个事件测试日志的补充。

根据测试用例总结中逐个测试的测试日志所提供的具体信息，我们可以得到如图 3-40 所示的测试套件总结工作表。左边往下你可以看到两列：测试套件名称和它所包含的测试用例数量。在某些测试日志的某个地方我们有每个测试用例的具体信息。在中间偏左的部分你可以看到在大标题“实现的计划的测试”下方有 4 列。这些测试已经没有任何相关未完成工作了，至少在这一次测试的通过中不再有任何未完成工作了。

表格中间一列是有关每个测试套件的加权失效数，它给出了有关测试套件发现多少缺陷的度量。我们用来自于先前工作表的缺陷风险优先级数字来对每个缺陷进行加权。所以，加权失效数字根据历史发现数据对每个测试套件的技术风险和发现问题的可能性进行度量。

在中间偏右的部分你可以看到在大标题“未完成的计划的测试”下方有 4 列。这些测试在这个测试通过中仍然有一些未完成的工作（顺便说一句，IP 是指正在进行）。

最后在右边我可以看到在大标题“获得的价值”下方有 4 列。获得的价值是一种简单的项目管理概念。它讲的是在项目中，我们通过消耗资源完成任务。所以，如果完成任务的百分比等于消耗的资源百分比，那么我们情况良好。如果完成任务的百分比大于消耗资源的百分比，那么我们倾向于低于预算完成项目。如果完成任务的百分比小于消耗资源的百分比，那么我们倾向于超出预算完成项目。

类似地，从进度的观点来看，完成任务的百分比也应当大致与项目过去的时间的百分比相同。和工作量一样，如果任务完成百分比高于进度百分比的话，我们会很高兴。而如果任务完成的百分比低于进度百分比的话，我们会难过且忧虑。

在测试执行中，可以考虑把测试用例或测试规程作为我们的基本任务。手工测试中的资源通常是指运行测试所需的人—小时数。这就是图 3-40 中按测试套件逐个显示获得

价值的方法。

套件	总用例数	实现计划的测试				加权失效	未实现计划的测试				获得的价值			
		总计	跳过	通过	失败		总计	已进入队列的	正在进行的	阻塞的	计划小时数	实际小时数	工作量%	执行%
功能性	14	14	0	4	10	10.60	0	0	0	0	36.00	49.00	136%	100%
性能	5	5	0	1	4	9.03	0	0	0	0	7.00	13.50	193%	100%
可靠性	2	2	2	0	0	0.00	0	0	0	0	0.00	0.00	0%	0%
错误处理	3	3	0	2	1	0.57	0	0	0	0	12.50	16.50	132%	100%
安装	4	0	0	0	0	0.00	4	4	0	0	72.00	0.00	0%	0%
本地化	8	1	0	0	1	0.50	7	0	7	0	128.00	22.00	17%	13%
安全性	4	3	0	2	1	2.10	1	1	0	0	17.00	13.50	79%	75%
文档	3	1	0	0	1	4.00	2	2	0	0	28.00	15.00	54%	33%
集成	4	4	0	3	1	1.00	0	0	0	0	8.00	12.50	156%	100%
可用性	2	0	0	0	0	0.00	2	0	2	0	16.00	0.00	0%	0%
攻击	6	0	0	0	0	0.00	6	6	0	0	12.00	0.00	0%	0%
总计	55	33	2	12	19	27.80	22	13	9	0	336.50	142.00	42%	58%
按百分比		60%	4%	22%	35%	N/A	40%	24%	16%	0%				

图 3-40 测试套件总结

通过在测试用例总结中获得的数据以及将其在测试套件总结中进行浓缩，我们就可以对测试项目进程构建出两种图形表现形式。第一种表示测试用例完成情况，如图 3-41 所示。

这个图有 4 项主要数据集，所有这些数据集都基于左边的单一坐标轴。第一个数据集是我们计划在给定测试通过中完成的测试数量，这部分显示为“闪电”。闪电中间的扁平部分是由于周末没有计划测试所产生的。我们会重置测试完成数据集，体现在图中就是每道闪电都从略大于零的部分开始，这是因为我们会运行一系列类似的测试通过，用重复测试来管理回归风险。

第二个数据集是每天结束时实际完成的测试数量，在图中显示为方框。第三个数据集是每天结束时通过的已完成测试的数量，在图中显示为灰色的加号标识。第四个数据集是每天结束时失败的已完成测试的数量，在图中显示为灰色的减号标识。我仍然使用标记来指出数据集有趣的方面，特别是为什么有些特殊的测试通过会比其他的更短。

图 3-42 是下一个我们可以从测试日志数据构建得出的图表，它代表了测试小时数的进展。这个图有两个主要的数据集，它们都基于左边的单一坐标轴。第一个数据集是计



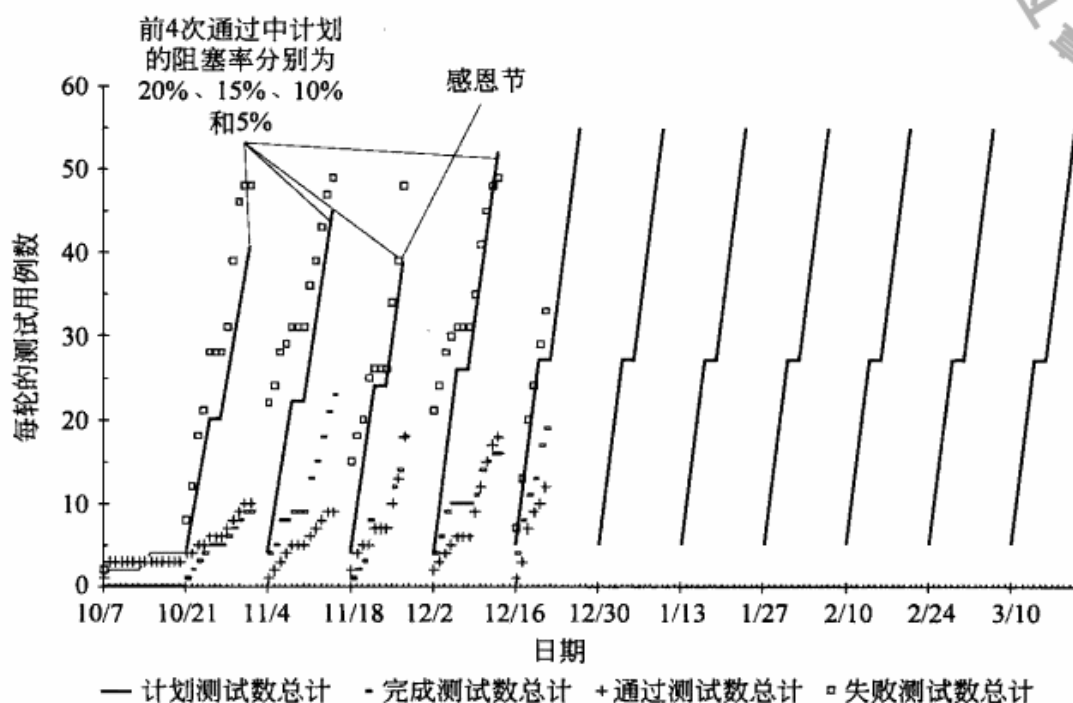


图 3-41 测试用例完成情况

划的每日测试执行小时数，在图中表现为横线。注意，项目中的小时数先是向上变化随后向下变化，这是指不同的计划的测试强度。第二个数据集是实际的每日测试执行小时数，在图中表现为方框。

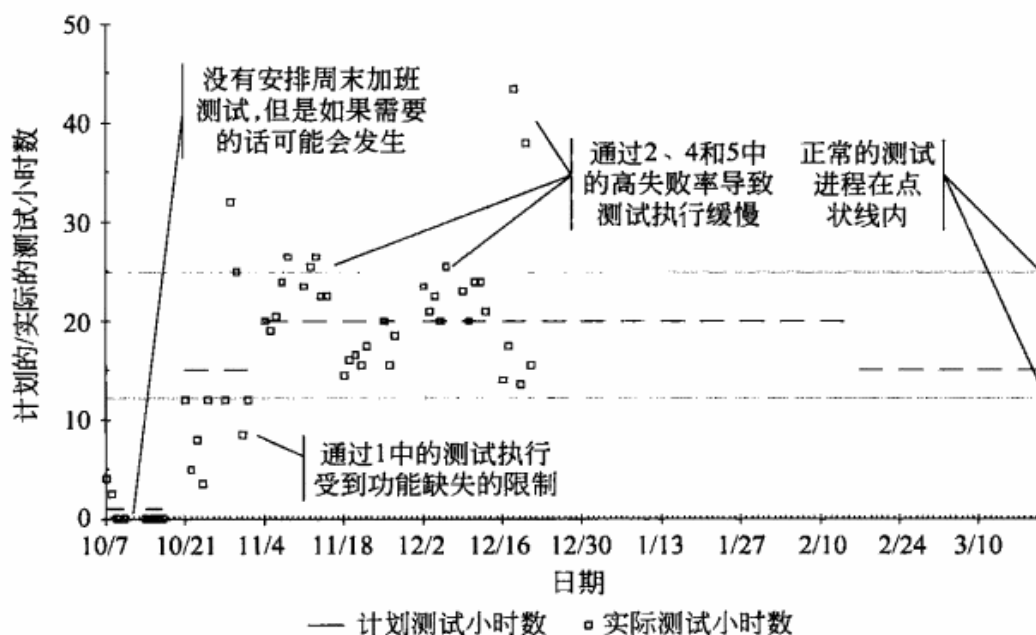


图 3-42 测试小时数

虚线上下方的两行浅灰色线代表了我们认为的正常的工作日进展情况。换句话说，只要方框标识仍然留在灰色线内，那么它们可以随机地在虚线上下方跳动。对于那些超出灰色线范围的方框标识，我们已经提供了相应的理由解释。

图 3-43 是确认测试的失效率，来自于缺陷度量分析。在我们的缺陷跟踪系统里，有一个计数器来记录对一个特定的缺陷一共打开过多少次。如果一个缺陷被发现、修复、

确认修复并且从此消失，那么这被记为打开一次。

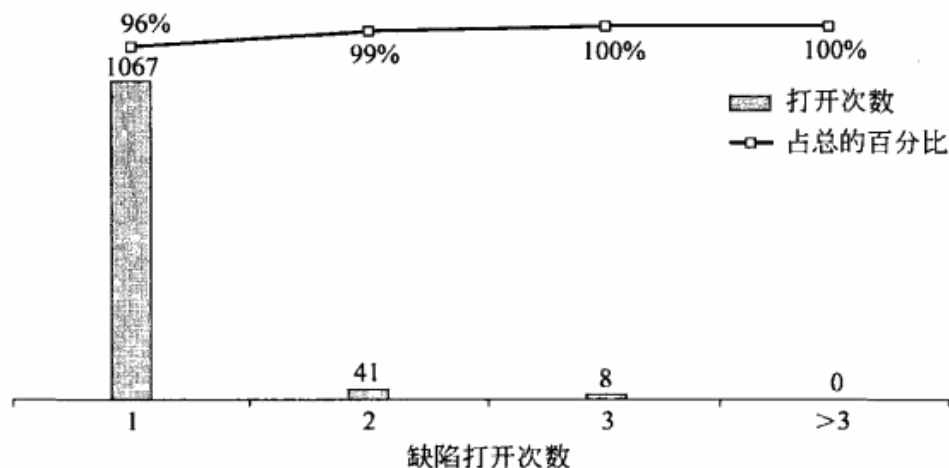


图 3-43 缺陷打开次数

但是，如果缺陷被发现且据称修复，但没有通过确认测试而被重新打开，然后真的被修复、确认修复且从此消失，那么这会被记为打开两次。额外的反复进行的重新打开的操作会进一步增加打开次数。

在这个例子里，我们仅仅对 4% 的已报告缺陷重新打开一次或两次。我们仅仅对 1% 的已报告缺陷重新打开两次，没有任何缺陷被重新打开超过两次，所以该项目在这个部分情况良好。

### 3.6.4 测试进度监控练习 1

接下来的图 3-44、图 3-45、图 3-46 和图 3-47 是 HELLOCARMS 测试迭代一结束时的缺陷和测试相关图表。根据这些图表中告诉你的有关 HELLOCARMS 测试的信息，为下一个迭代列出 3~5 项改进措施。如果这些改进措施奏效的话，迭代二结束时的图表会是怎样的呢？

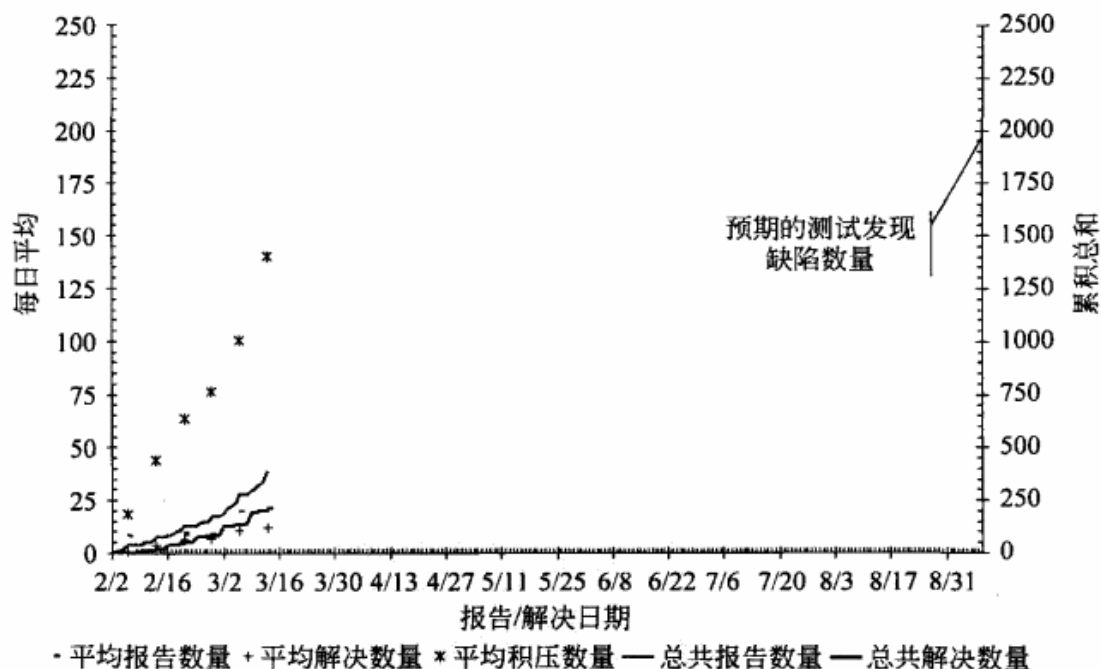


图 3-44 迭代一结束后累计报告和解决的缺陷



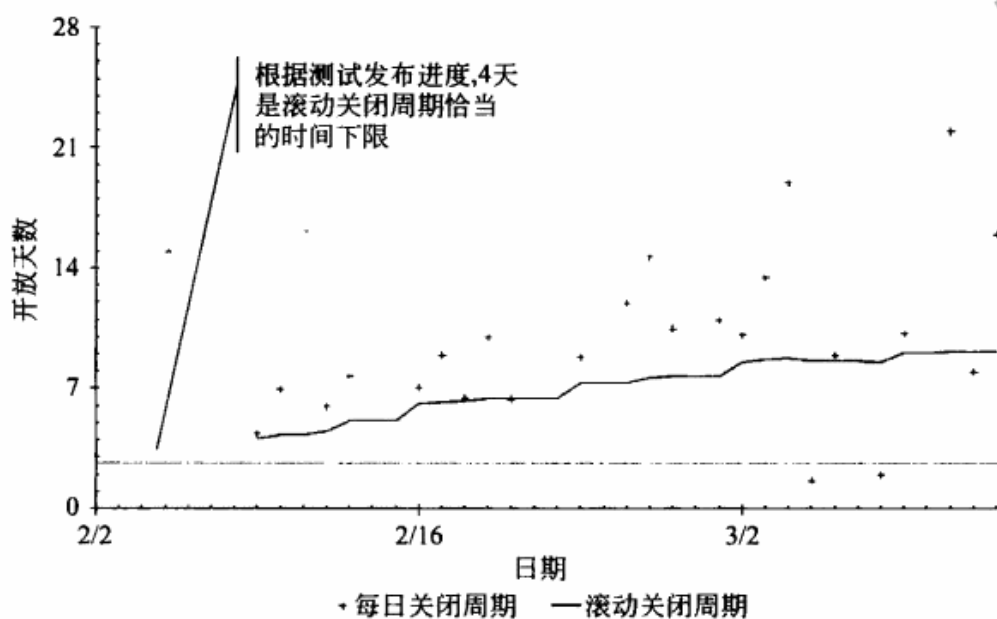


图 3-45 迭代一结束时的关闭周期

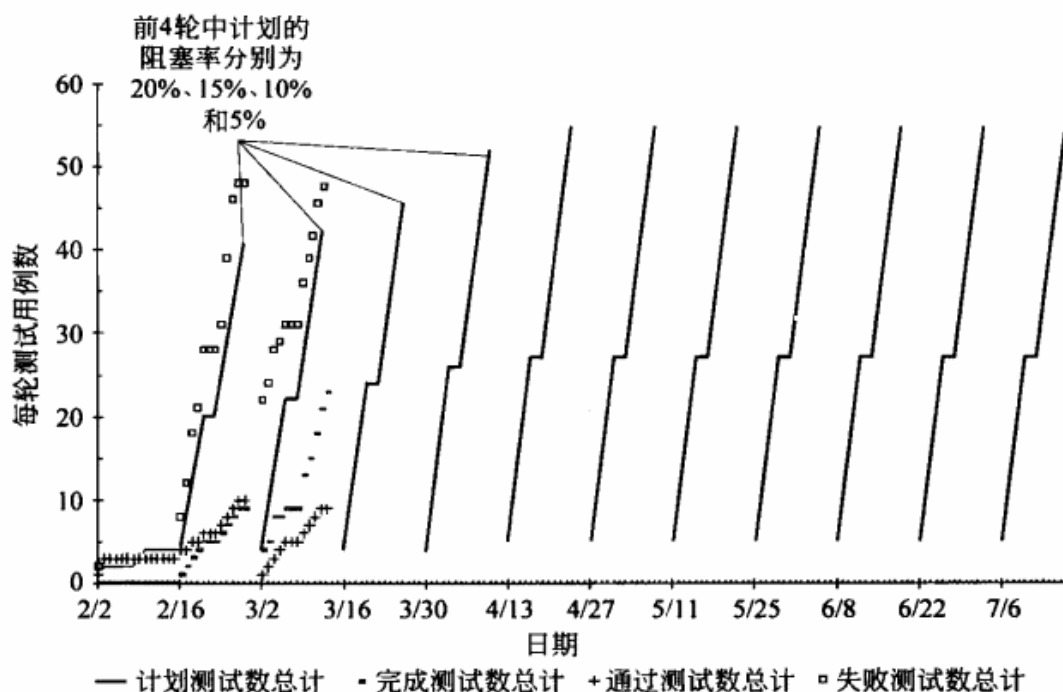


图 3-46 迭代一结束时的测试完成情况

如果在教室里学习，请将大家分成3~5个小组。如果您是单独学习，需要独自完成这个练习。如果在教室里学习，当每一个小组完成了他们的分析后，对分析结果进行讨论。

我建议用45分钟来完成这个练习，包括讨论的时间。

### 3.6.5 测试进度监控练习1 参考答案

看了这些图以后我注意到4个可以进一步调查和可能改进的地方：

1. 在测试阶段刚结束时累积报告/解决缺陷图有一个大的跳动。

可能的改进：在迭代中更早地稳定产品，本例大约在第三周左右。另外专注于修复

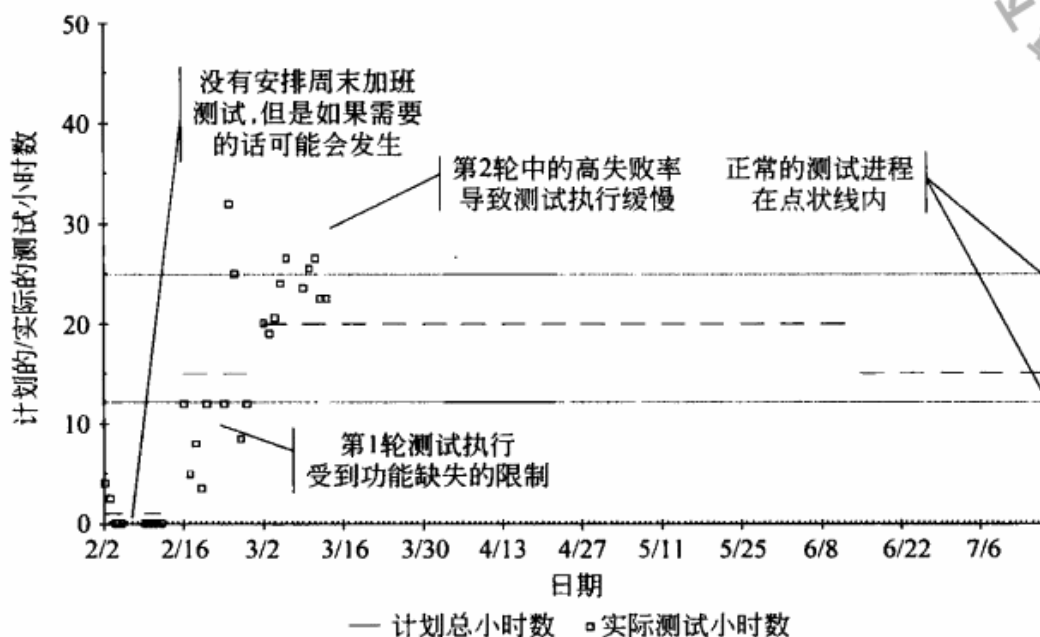


图 3-47 迭代一结束时的测试小时数

在该迭代的后半段测试中发现的缺陷。

2. 关闭周期图表在整个测试的关闭周期都显现稳步向上的趋势。

可能的改进：根据新的功能限制后续迭代的规模并且专注于快速消除缺陷。

3. 测试用例完成情况图中在迭代的第二次和第三次（例如，最后两次通过）通过中增加了大量的测试，同时也使得后一次通过中有大量的测试失败。

可能的改进：尝试在第一次通过中运行所有的测试从而识别到尽可能多的问题；然后专注于缺陷的解决从而使迭代中最后一次通过结束时大部分的测试可以通过。

4. 在测试小时数图中可以看出在第一次测试通过中的测试牵引有限，造成在第三次和最后一次通过中测试进度压力。

可能的改进：移除任何对有效测试和首次通过中的测试小时完成情况有障碍的部分。

你可能已经基于这些图表以及为了理解问题的原因而进行的进一步研究，也已经注意到其他可能的改进。

### 3.6.6 测试覆盖率度量

测试覆盖率度量没有测试用例度量和缺陷度量那么常用。一种可能最常见的覆盖率度量是需求和设计覆盖率度量。如果要生成这样一个度量，你需要从测试追踪到它们覆盖到的需求或设计。例如，测试用例和测试基准中的每一项都是双向的。测试用例、可追溯性信息和测试基准文档的粒度必须足够细从而保证可追溯性有意义。例如，如果你有一个单独的文档标记为“系统需求”，上面有一些随手画的图，另外你有一个探索性测试纲要集合，其中每个纲要描述了测试基准部分中的系统需求，那么这无法提供有意义的追溯性，从而你也无法得到有意义的覆盖率度量（我将会在本章的稍后部分讨论探索性测试纲要）。

另一种类型的覆盖率度量是风险覆盖率度量。你在这部分开始的时候看到过一个包括了暗灰色、中等灰色和浅灰色部分的饼图，这就是风险覆盖率度量的一个例子。再次



强调的是如果想要这个图能够正确工作，需要确保测试用例和质量风险项间具备有意义的和足够详细的可追溯性。

还有另一种类型的覆盖率度量关注的是环境或配置覆盖率。这种类型的度量对于那些用收缩性薄膜包装的软件和其他大众市场软件或客户在许多环境下使用的系统特别适用。您需要为每一个重要的环境因素捕捉为测试所选的选项。例如，如果应用程序运行在 Windows 个人电脑上，可以捕捉具体的 Windows 版本和补丁包版本。在某些情况下，组合的问题诸如选项两两组合或三三组合等都很重要，但是通常您会希望避免陷入组合的问题。关于这个组合测试的话题会在本书的姐妹篇高级测试分析师考试篇中进行讨论。

最后一种测试覆盖率度量是代码覆盖率度量。在基础级大纲第 4 章中已经讨论过这种代码覆盖率度量的通常使用类型。和迄今为止其他讨论过的覆盖率度量形式不同的是，代码覆盖率几乎从不通过可追溯性表格或数据库进行度量，它大都通过工具来追踪测试用例运行时累积的总体代码覆盖率。

因为这些度量帮助您作为一名测试经理来了解所需测试的内容是否已经被测试或将会被测试，所以这些度量十分重要。在已经根据需求元素、风险项或配置运行的测试用例中，可以用可追溯性来追踪通过、警告、失败或阻塞这 4 种测试结果，从而追溯到尝试测试和报告其状态的基本项。

最后，注意这些覆盖率度量可以作为信心度量的替代品。我们可以通过对系统测试覆盖率的程度和覆盖到的测试的实际运行情况的了解粗略地度量出我们对系统所具有的信心级别。

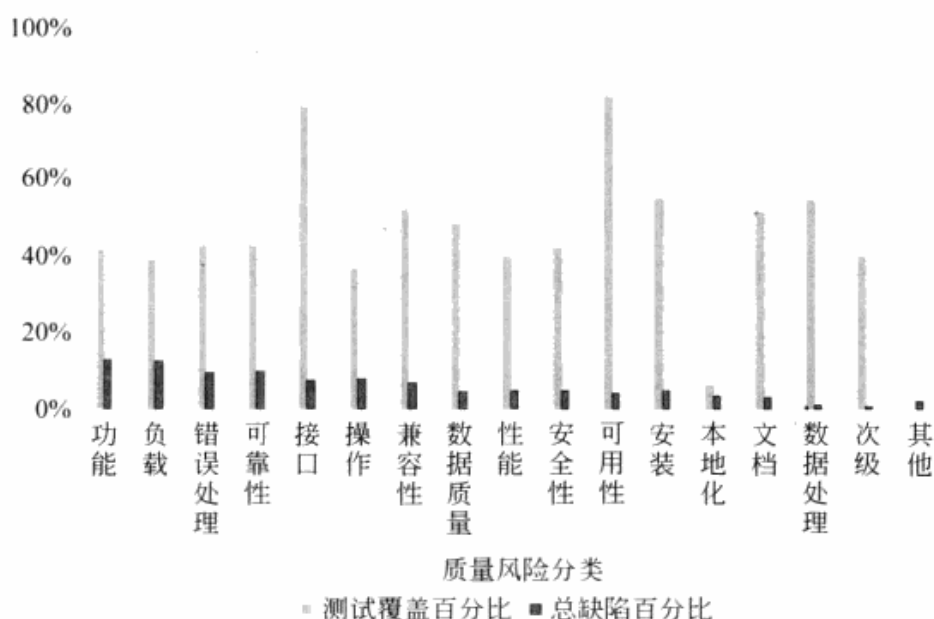
当组织运用覆盖率度量作为信心的替代度量时可能会犯一些错误，例如依赖于单独的因素进行度量，例如需求覆盖率或代码覆盖率，而不考虑其他的度量覆盖率。理想状态下，在决定是否应该对系统有一个高级别的信心前，你会运用多因素的覆盖率度量，对代码覆盖率、需求覆盖率和风险覆盖率进行检查。对于任务关键和安全关键系统，使用多因素的覆盖率度量是一种聪明的做法。

### 3.6.7 风险覆盖率

图 3-48 分析了各个关键质量风险类别之间的关系，每个类别已经达到的风险覆盖率程度以及已经发现的与每个风险类别相关的缺陷数量。

在底下，我们可以看到与每个主要风险类别相应的标签。我们需要获得从测试用例到它们相应风险项的可追溯性。我们用风险类别级别的可追溯性比较粗的颗粒度来创建这个图。我们也需要根据缺陷相关的风险来对我们的缺陷报告进行分类，同时我们也会用这种可追溯性来生成这个图。如果我们有更细粒度的可追溯性，比如在特定的测试用例和风险项之间以及在特定的缺陷报告和风险项之间，那么我们可以为每个质量风险类别生成一个具体的报告，这很可能是用表格的形式，针对每个风险项显示出通过的测试、失败的测试、已解决的缺陷和未解决的缺陷。

我们用测试覆盖率可追溯性信息在每个类别中划分已达到的计划风险覆盖率百分



比，在图中显示为灰色的条形。随着灰色的条形逐渐成长到 100%，这个类别中的剩余风险数量会不断下降，因为我们还没有测试的部分越来越少。当到达 100% 时，我们已经实现了对该类别的计划风险的缓解工作。假如我们的风险分析做得很好，没有任何其他未知风险遗漏，那么该类别中唯一的剩余风险是与已知缺陷有关的风险。注意我们没有计划针对“其他”部分的测试，这是因为我们用这个风险类别来包含那些我们发现的无法追溯到已识别风险的缺陷。如我在本章前面提到的那样，这个类别可以帮助我们在测试执行期间检查我们的质量风险分析工作。

我们用缺陷覆盖率可追溯性信息在每个相关类别中划分缺陷百分比，在图中显示为深灰色的条形。这些条形一个个叠加在一起通常总和将会达到 100%。如果缺陷条和测试条相比更高的话，那么这代表有重大的风险。这是因为虽然我们才运行了少量的测试就已经发现了大量的缺陷。相反的，如果缺陷条和测试条相比更低的话，那么这代表风险不高，这是因为我们已经达到了高测试覆盖率但是却没有发现许多缺陷。如果“其他”类别中的缺陷条超过 5%，我们应该开始对我们的质量风险分析做出怀疑。如果超过 10% 或者说其中缺陷数量超过所有的地方所发现缺陷的 1/10，这包括我们并未预料到会发现许多缺陷的地方以及我们认为缺陷不会很重要的地方，那么我们绝对应该重新修正我们的质量风险分析。

### 3.6.8 度量的使用

可以通过各种不同的形式来使用所有这些度量。可以用口头叙述的方式，可以用具体的或概括的数字表格的方式，也可以用可视图形的方式。可以根据度量的目的和度量使用的目的来决定恰当地使用形式。

- 是否打算分析项目、过程或产品从而发现或强调某些特定的担心的地方？
- 是否打算向有兴趣的项目和产品参与者和利益相关者报告你的测试结果？
- 是否打算依据度量生成控制管理的决定从而调整整体的测试或项目进程，并且监



督进程修正的结果？

作为一名测试经理，必须记住如果这些度量的接收者是除您之外的任何人，那么正确地收集、分析和报告将会随读者而定。您应该考虑到他们的需求、目标和能力。<sup>15</sup>

例如，如果想发送未解决缺陷的信息，那么您会给开发主管和市场经理发送不同的报告。经理们的需求有别于技术人员的需求。

另一个例子，如果想要发送未解决缺陷的信息，您可能会给项目经理和企业方项目赞助商发送不同的报告。项目经理可能需要具体的未解决缺陷的信息，但是商务经理会倾向于了解产品风险的状态。

如果我们认为测试度量指出了某类问题，我们有各种选择来解决问题并且将项目重新带回可控状态。我们可以修订测试路线图，例如质量风险分析文档、测试计划或测试优先级。例如，如果我们发现项目最后期限已经发生变化并造成测试无法完成，我们可以根据风险分析降低测试覆盖率并且向下调整相关的风险项优先级。

我们可以增加测试资源或总体的测试工作量。例如，如果发现我们缺少具备足够技能的人员或系统来进行性能测试时，我们会将测试外包。

我们也可以决定延迟发布时间来进行更多的测试。很少有人主动选择这种方式，通常只会在没有其他选择时采用这种方式。

我在许多项目上经常看到的决定是在发布日期或其他项目的重点受到威胁时改变测试的退出准则。例如，你可以放宽在最后两周缺陷发现率需要达到零的这个标准，特别是当发现的缺陷数量很少并且缺陷的重要性较低时。在某些情况下，退出准则可能会被加强，但是我不得不说我还从来没有遇到过这种情况。

虽然测试经理可以也应该在报告测试结果时概述项目团队可选择的办法，并且对应该选择哪种办法提出建议，但是实际上选择和实施一种或多种方法通常超出了测试经理的职权范围。通常这需要一个或多个产品和项目利益相关者及经理达成一致或至少在大方向上达成共识。作为一个测试经理和咨询师，我已经发现做出片面的决定从而影响公司战略，会导致当事人不得不在其他的公司寻找一份新的工作。

图 3-49 是一个用来监控测试进程的测试套件总结的例子。您会注意到这个图是前面看到的测试套件总结的简化版，这是我已经提到过多次的 IVR 项目的测试套件总结，这是在客户服务应用程序（CSA）的系统测试中得到的。您应该已经在本章前面部分看到过这个系统的失效模式和影响分析的例子。

在测试套件总结中，您应该会注意到大量的阻塞和失败的测试。牢记，阻塞的测试是那些我们想要运行但无法运行的测试。这时候，如果能让测试重回正轨，很重要的是让网络运行团队解决阻塞的问题（大部分是环境的问题）并且让 CSA 开发人员修复那些导致测试失败的缺陷。

---

15 有 3 本书可以帮助您学习更好地使用度量并且更好地进行表述，它们是 Edward Tufte 的 “The Visual Display of Quantitative Information”、“Envisioning Information” 和 “Visual Explanations”。我发现读了这些书以后并且将 Tufte 的想法付诸行动之后极大地改进了我分析和表达想法的能力。

系统测试 2.5 (CSA) 测试套件总结									
套件	总用例数	阻塞	完成	失败	加权失效	通过	在处理中	跳过	排队中
压力和容量	16	4	12	2	3.44	8	0	2	0
性能	8	1	7	7	1.00	0	0	0	0
错误处理/恢复	18	5	13	7	1.53	5	0	1	0
数据质量	26	4	22	9	4.53	13	0	0	0
功能	23	1	22	7	1.06	14	0	1	0
	91	15	76	32	11.56	40	0	4	0
		16%	84%	35%	N/A	44%	0%	4%	0%

图 3-49 例子：测试进程监控

在 IVR 项目里，有严重问题的不仅仅是 CSA 应用程序。用来处理客户来电的 IVR 应用程序违反了系统测试入口准则，并且因此在系统测试期间遇到了许多的问题。在系统测试的中期我写了图 3-50 中的这封邮件给项目经理，你可以看到我建议根据缺陷报告的度量考虑延迟系统测试的结束和系统集成测试的开始时间。

12 月 12 日，我（作为测试经理）写给项目经理：

【项目赞助商、开发经理和我】已经就【开发经理】强调的至今为止 IVR 应用程序测试中发生的滞后情况进行了讨论，我们很可能会将 IVR 应用程序测试完成的时间延迟到 2 月 28 日。【项目赞助商】不想听到这个……虽然我们没有达到入口准则，但我们还是在 11 月 23 日进入了系统测试。在过去的 3 周，情况变得越来越明了，我们至少早了 4 个星期进入系统测试……【根据至今为止的缺陷度量】到了 1 月 4 日我们将会得到一个几乎垂直的【累积缺陷报告】曲线……如果这样的话，我们不应当【按原计划】进入【系统】集成测试……

12 月 12 日，项目经理写给我：

您的担心是对的……但是让我来解释一下为什么进度表不能改变。【客户】付了一大笔年底奖金并且他们坚持维持【现有的项目进度表】……【因此】我们必须找到一个办法来弥补不足。我计划在 1 月 1 日结束系统测试【并开始系统集成测试】……让我们在当前进度要求下尽全力工作。

图 3-50 调整测试过程

项目经理的这种回答十分常见。项目利益相关者的奖金有时会和项目进度挂钩，这些奖金会引入极大的要求符合进度的政治压力，并且这也是为什么项目赞助商“不想听到”IVR 应用程序测试的问题的原因。

因此项目经理在他的回答中宣布进度是首要的，所有人必须保证测试开始和退出的时间，于是我的测试团队需要在这些限制条件下进行工作。没问题，这是很清楚的指示。我对这个决定并不高兴，但是它确实清楚地指明我们必须在不影响到进度的前提下处理那些让测试阻塞或失败的问题。

这封邮件的内容是否表明项目经理和项目赞助商是坏人呢？如我所说，我在那时对



这种情况并不高兴。但是，重新思考后我发觉他们的反应都是完全理性地出于公司董事会所设置的财务动机。通过设立和进度完全绑定的奖金，严格设定预算的上限以及通过进一步的没有奖金或没有与质量相关的验收标准或甚至没有与功能完成度相关的验收标准，董事会为自己设定了最终发生的结果：每一个尝试都是为了将产品匆匆投入市场，包括满足于有限的特征和功能以及推迟修复大量的质量问题，其中有一些质量问题还相当严重。

这种尝试失败了，最终大部分高级项目利益相关者都遭到了解雇。具有讽刺意味的是，其中一个提供了含有大量缺陷的组件的外部承包商不仅最终仍然被公司保留了下来，而且他们中一人还在高级项目利益相关者被解雇以后接手成为公司负责承包的高级职员，负责他们的技术运营。生活有时候就是这么滑稽。

那么当测试的现实情况和我们精心准备的测试计划有所偏离的时候我们该怎么办呢？作为测试经理，我们必须定义和执行一项或多项测试控制。我们想要尝试在给定的限制条件下达到可能的最佳结果。有哪些测试控制度量可供我们使用呢？与项目控制一样，两者相当类似。

您可以重新审视测试用例的优先级或甚至取消某些测试。您可以根据风险分析或限制条件来进行以上活动。如果运行特定的测试所需的硬件不可用，那么不妨取消这个测试。

如果现实的偏离是您已经预料到且做过相应计划的，那么您可以实行在测试计划中定义的应急措施。

如果您的预算有灵活性而进度是固定的话，可以尝试获取额外的资源。相反的，如果您的进度有灵活性但是预算固定的话，可以尝试推迟测试结束时间或发布时间。但是不要忘记培训的时间和成本。

如果进度和预算都不能改变，那么可以尝试改变项目整体的范围或仅仅改变测试的范围。

最后，可以尝试修改测试退出准则来使其更容易“宣布成功”。

和项目控制一样，您的决定需要一个或多个项目利益相关者、经理或赞助商的一致同意、准许或批准。小心不要在没有得到足够批准之前，从某些测试任务、测试覆盖或其他你的项目责任中悄悄撤离。当你在星期五晚上工作得很晚的时候会容易这么做，但是记住不要向诱惑屈服。

### 3.6.9 两个测试报告的案例学习

图 3-51 是我发给两个客户联系人有关因特网装置项目的邮件。问题是这样的：我们还没有收到必需品、服务器和客户端这些主要产品部分中任何一部分的足够详细的需求规格说明，这导致我们无法开发我们所有相关的测试。同时，这也造成了在非正式测试执行开始时，我们的测试设计和实现会有一定的延迟。所以，在这封邮件里我建议了一种根据质量风险对测试用例开发进行挑选的方法。我们会根据关键程度的高低来决定测试用例的严格和详细级别的高低。

大家好：

我们正面临一个严重的风险那就是无法完成【测试】开发。即使“问题”文档中的每个问题【例如，需求中的未解决问题】都在今天得到解决，基于测试人员现有的大量的测试执行需求，我仍然怀疑【测试工程师】能有时间来完成所有未完成测试用例的开发。【因此】我建议我们（我们3人）用以下类别来挑选剩余的测试用例：

1. 覆盖关键的质量风险。必须在这部分寻找尽可能多的缺陷；必须完成测试开发并生成可靠的测试用例。
2. 覆盖重要的质量风险。必须尝试发现关键的（影响客户的、数据丢失或商务妥协）缺陷；必须完成测试开发，但是测试用例可以留有一定的不清晰性。
3. 覆盖非重要的质量风险。在这部分进行随机测试或不进行测试都是可以接受的，在时间允许的情况下完成测试开发。

我们可以和所有相关人员召开【一次】会议……【来解决】所有和第一类测试用例有关的问题。

图 3-51 改进测试过程

在这封邮件的回复中，其中一个客户联系人要求提供一些有关未完成测试数量的度量。我能够提供这些细节。根据这些信息，我们能够继续进行测试完成阶段。但是我在邮件结尾处提到的会议最终并没有召开。取而代之的是我们被告知根据现有的信息尽我们的所能做到最好，所以我们在测试规格说明书和测试执行中采用了“最佳猜测”的方针。这种方针导致了在测试执行期间存在大量的错误肯定，这实际上把缺陷报告过程变成了反向追溯需求规格说明的过程。

表3-17是因特网装置项目接近系统测试结束时我们的测试结果总结实例的一部分内容。确切地说，这个表格是我们测试团队必须修复缺陷列表报告的节选。我们的两位客户联系经理要求我们列出我们认为会影响到系统交付的严重问题列表。

表 3-17 测试结果总结举例

缺陷号	问题	质量影响
153, 2 259	缺少 Java 支持	一些网页上最酷的部分【无法工作】。还看到其他一些有关商城的缺陷……
825	装置移动处理得不好	【供应】过程应当在我们正式上线之前就位，这落后于客户软件发布多远了呢
43, 984, 1 934, 2 191, 2 510	没有工作的音频……	我们有扬声器。客户将会希望它们可以工作
1 145, 1 405, 2 417, 2 543	安全性差	在我们把这个系统发布到这个世界之前，就好像把羊送入狼群之前，它需要是安全的。如果这个产品成功了，竞争者和黑客将会测试其安全性
1 204, 1 479, 2 056, 2 330	拨号上网不可靠	【拨号上网在 Austin 以外工作得不好。】我们担心这个服务不会变好
1 252	无法改变用户的电子邮件地址	结婚、离婚、性别变更？无论什么理由，人们应该能够改变他们的电子邮件地址

通常而言，作为测试经理，我倾向于不生成任何类型的“必须修复”缺陷列表。我相信这项工作最好应该由项目利益相关者的跨职能团队去进行。但是，在这个例子里，



客户要求我们给出意见，所以我们给出了我们的想法。我们和这两个经理有良好的关系，因此我们可以在报告中使用一些幽默的语言。我已经删除了邮件中另外的大约 15 行内容，因为它们和理解我们在做什么以及为什么要这么做关系不大。

注意这并不是直接来自于缺陷跟踪工具的未加工的报告。我们已经在一定程度上对问题进行了分析来寻找常见的失效模式，然后对这些失效模式对客户的影响进行评估。详细内容可以通过回顾具体的交叉参照的缺陷报告获得。

### 3.6.10 测试进度监控练习 2

接下来的图 3-52、图 3-53、图 3-54 和图 3-55 是练习 1 中同样的缺陷和测试相关图表，但是时间是迭代二测试最后一周开始的时候。分析这些结果并回答以下两个问题：

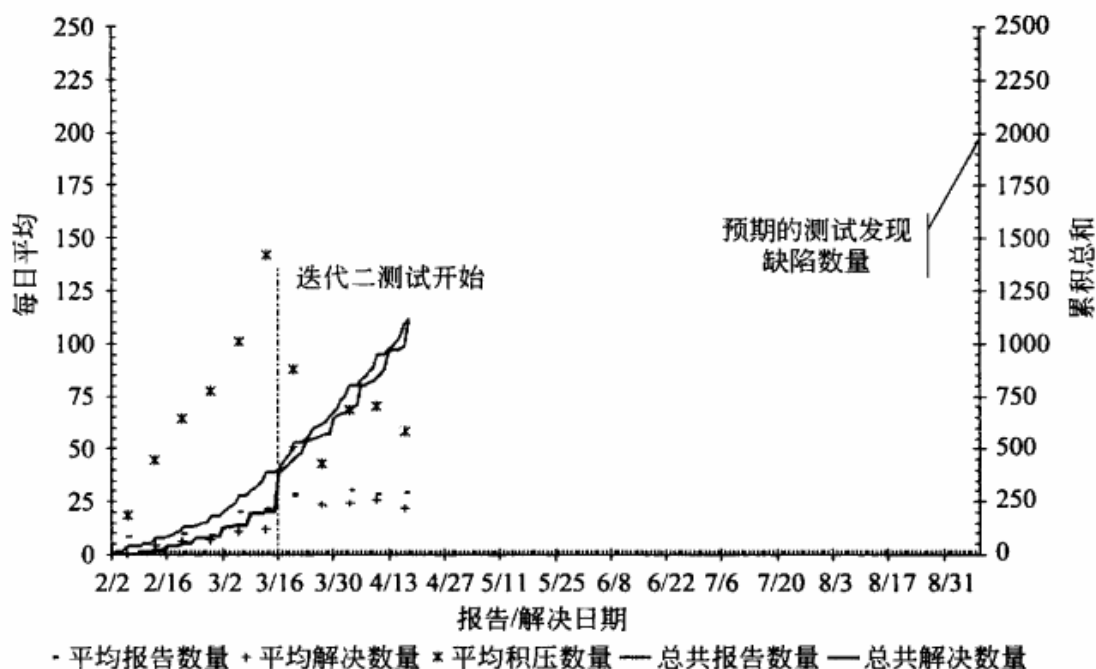


图 3-52 迭代二接近结束时累积解决的缺陷报告

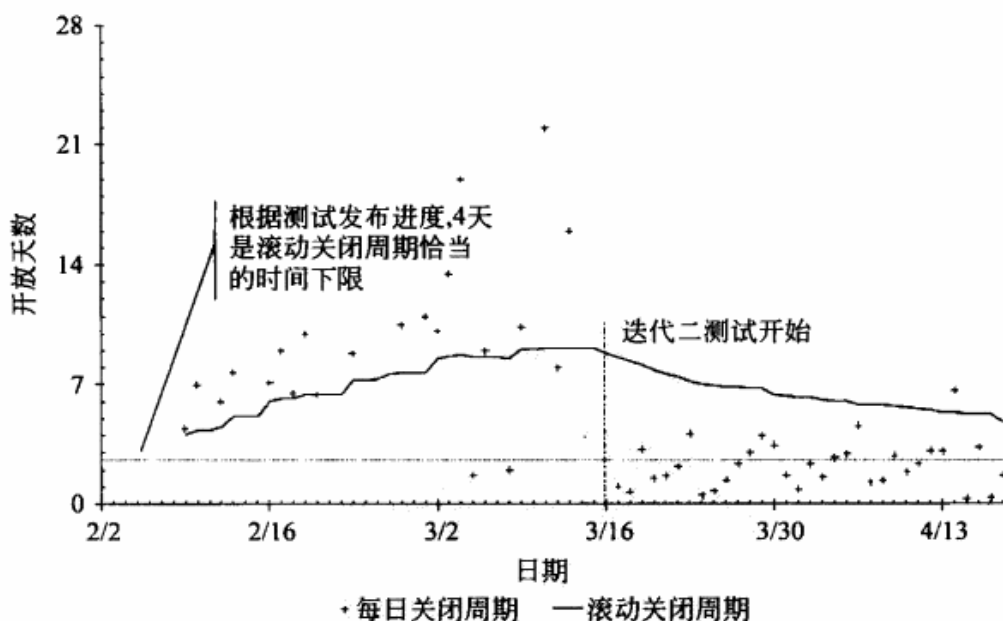


图 3-53 迭代二接近结束时的关闭周期

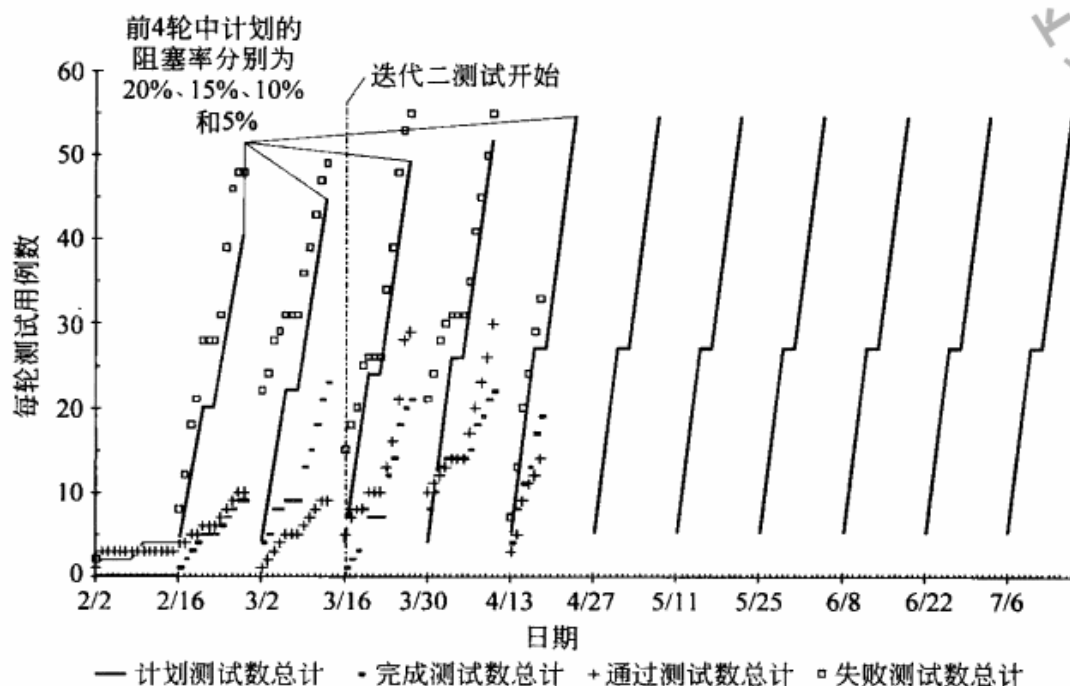


图 3-54 迭代二接近结束时的测试完成情况

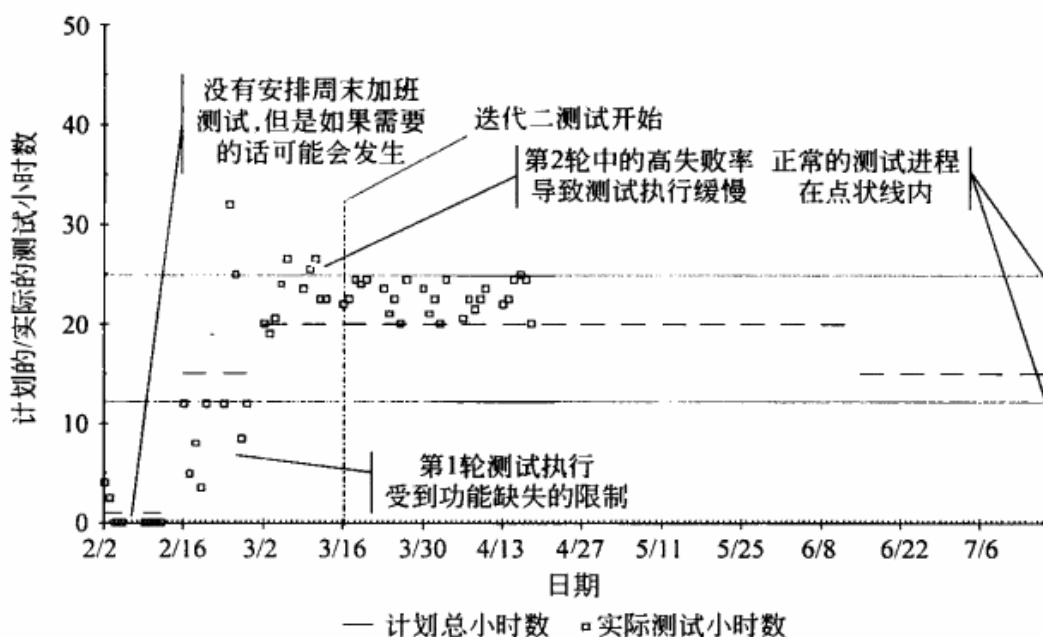


图 3-55 迭代二接近结束时的测试小时数

1. 图中是否有证据显示你在练习 1 中所概述的改进已经奏效?

2. 你是否相信迭代二中 HELLOCARMS 的测试结果表明其在迭代二结束的时候可以准时交付?

如果在教室里学习,请将大家分成 3~5 个小组。如果您是单独学习,需要独自完成这个练习。如果在教室里学习,当每一个小组完成了他们的分析后,对分析结果进行讨论。

我建议用 60 分钟来完成这个练习,包括讨论的时间。

### 3.6.11 测试进度监控练习 2 参考答案

让我们根据我在练习 1 中建议的 4 项改进开始对进度进行评估。



1. 在测试阶段刚结束时迭代一的累积/解决缺陷报告图有一个大的跳动。

可能的改进：在迭代中更早地稳定产品，本例大约在第3周左右。另外专注于修复在该迭代的后半段测试中发现的缺陷。

迭代二的状态：虽然开发确实已经在解决缺陷方面做得很好，在迭代二中已报告缺陷总数也几次达到了收敛状态，但是缺陷报告率在迭代二的整个测试执行阶段中都非常高。测试经理和项目管理团队应当一起进行进一步的调查来寻找这个问题的根本原因。这有5种可能的原因：

- A 迭代一中功能的高回归率是因为迭代二中的高缺陷修复率
  - B 迭代一和迭代二中新缺陷的高发现率是因为测试范围的扩大
  - C 迭代二中新功能增加率高
  - D 迭代二中增加的大量功能带来了大量的缺陷
  - E 上游质量控制活动不够，例如需求和设计评审、代码评审和静态分析、单元测试
2. 关闭周期图表中缺陷关闭周期在第一轮测试周期显现稳步向上的趋势。

可能的改进：根据新的功能限制后续迭代的规模并且专注于快速消除缺陷。

迭代二的状态：我们无法断定迭代二的规模是否已经受到限制，因为这是迭代二中缺陷发现率高的一种可能的解释。但是，开发团队确实已经做得很好，他们扭转了工作积压和缺陷关闭周期延长的负面趋势。换言之，根据上面的描述可能开发团队工作进行得太快且解决缺陷时对细节关注不足，这导致了高回归率。

3. 测试用例完成情况图中在迭代一的第二轮和第三轮（例如，最后两轮）中增加了大量的测试，同时也使得最后一轮中有大量的测试失败。

可能的改进：尝试在第一轮中运行所有的测试从而识别到尽可能多的问题；然后专注于缺陷的解决从而使迭代中最后一轮结束时大部分的测试可以通过。

迭代二的状态：测试团队做得很好，他们加快了测试完成的速率以及每轮通过中的总体测试完成数量，但是我们仍然看到有大量的测试失败。在最后一轮中这个问题显得尤为尖锐，在最后一轮中大部分测试都失败了。当然，这和迭代二累积/解决的缺陷报告图所示的结果是一致的。

4. 从迭代一的测试小时数图中可以看出，在第一轮测试中的测试牵引有限，造成在第三轮和最后一轮中测试压力很大。

可能的改进：移除任何对有效测试和完成首轮测试有障碍的部分。

迭代二的状态：测试团队在确保完成牵引和在迭代二测试执行一开始就切入测试方面做得很好。迭代二的实际测试小时数比计划的要多一些。这部分时间和迭代二测试用例完成情况和累积缺陷报告/解决图所示的结果是一致的，因为预料之外的高测试失败率和高缺陷报告率会增加所需的测试工作量。

因此，根据这些分析，我们可以得出结论：HELLOCARMS 在迭代二结束的时候部署并未进入正轨，累积/解决缺陷报告图表明系统中还留有大量的潜在缺陷。测试用例完成情况图则表明大量的功能存在问题。

## 3.7 测试的商业价值

### 学习目标:

(K2) 列举出决定质量成本的 4 类别中任一类的实例或者度量。

(K3) 对于某个指定的上下文, 列出用于测试的数量以及/或者数值。

在测试领域中, 有很多人, 甚至是一些我敬佩的人, 比如说 Boris Beizer, 他曾经说过, “不要试图为测试构建一个商业用例。”他们认为这是在浪费时间, 因为无论如何经理们也不会采纳这个构建的结果。但我的经验告诉我, 事实并非如此。我发现传统且解释清晰的商业用例可以说服并影响项目利益相关者。同时, 我还发现改变如何看待测试的观点也很重要。

在测试世界中存在类似于美国社会的宗教组织问题。根据社会学调查, 美国的无神论者数量非常少, 即他们通常肯定地认为没有神灵。还有一些不可知论者, 他们不能肯定是否有神灵, 但是他们中的大部分人说他们相信某些神灵或者其他类似的东西存在。

但是, 如果你问这些不可知论者, “你上次去清真寺、犹太教堂、基督教堂、庙宇, 或者其他朝拜地是在什么时候?” 答案可能大多数都是“我堂兄的婚礼”、“我叔叔的葬礼”、“我侄子的成人礼”, 而你很少会听到“我每个星期都去”, 更少会听到“我每周去 3~4 次”。也就是说, 参加宗教组织仪式已经不是很多人优先考虑的事。在美国很多宗教派别的财政状况也因此非常糟糕。美国人很多都是被动接受宗教。<sup>16</sup>

测试也是这个样子。认为测试无用论者, 也就是认为测试没有任何价值的人非常少。大多数人会说测试有一些价值。但是, 当你问他们, 为了减少风险、长久地节约费用, 增加信心以及测试可以提供的其他信息, 他们愿意付出什么, 可以承担多大的延迟, 加强哪些特性。你很快就发现大多数 IT 人士对测试其实和对宗教一样都是被动接受。

我认为这个问题主要是由两个原因引起的。首先, 很少有经理, 甚至是熟练的测试经理, 可以量化地描述或者清楚地将测试的价值传达给同级别的经理、高级经理, 或者执行领导。其次, 很多测试人员乐于在他们得心应手的领域工作, 处理一些技术上的问题, 例如: 寻找缺陷、撰写测试用例, 以及建立测试环境。他们常常忽略了更深层的策略问题, 而这些策略问题可以将测试人员的行为与决定公司价值的重要事项联系起来。

所以, 我建议, 作为测试人员, 要遵守以下 3 个商业价值测试的原则:

- 首先, 保证完成测试所需的度量事项。
- 其次, 保证对有价值的事项提供充足资金。
- 第三, 保证测试行为具有可度量的价值。

如果能接受以上 3 个原则, 那么作为测试人员所做的事就和公司真正关心的事项紧密联系起来, 这样也就很少会有资金上的困难。

运行测试从本身来讲, 没有任何价值。这个听起来好像和我刚才说过的话有点自相

16 具体的例子, 请查看 [religions.pewforum.org/reports](http://religions.pewforum.org/reports)。



矛盾，但其实一点都不矛盾。我的意思是执行测试，例如：敲键盘、读数据库、观察屏幕的显示，以及检查报告等，这些活动在本质上没有任何价值。只有当测试和公司目标或者其他目标相结合时，测试才有价值。

有些测试的目标是可以量化或者说是可以计量的：

- 如果公司想减少长期的缺陷相关成本，我们可以在产品发布前帮助其找出必须解决的缺陷。
- 如果技术支持或者服务人员让始料不及的问题搞得疲惫不堪，我们可以在发布前找出可延迟且不是那么关键的缺陷，以帮助他们对这些问题事先有所准备。
- 如果公司担忧发布的软件的相关风险，我们可以通过运行测试来减少风险，不管这些测试能否发现缺陷。
- 如果公司要在决定版本发布与否时更有信心和根据，我们可以帮助提供相关信息。

而有些目标是定性的，但对掌握这些目标的人来说，目标却是非常的真实：

- 如果公司想通过质量的改善赢得声望从而来促进市场和销售上的增长，我们可以在整个生命周期中给予帮助。
- 如果公司想得到更加顺利的可预测的发布周期，我们可以参与整个生命周期并给予最好的帮助。
- 如果管理团队想要增加版本发布时的信心，我们可以通过和他们一起工作来保证最重要的部分已经进行了很好的测试——比如使用基于风险的测试。
- 如果公司由于规则或者责任问题需要得到法律责任上的保护，我们可以通过确保合规性的保护性测试来给予帮助。
- 如果对于安全关键或者任务关键的业务，公司想要确保任务失败或者造成人员伤亡的发生几率为最小，我们可以在整个生命周期中和项目团队合作，在发布前筛选出尽可能多的问题。

当前，对你的项目，对你的公司来说最重要的是什么？如果你不能很好地回答这个问题，那么你也不可能致力于正确的事情。测试经理应当知道选用哪些定量和定性的价值，而且能够将这些价值和项目利益相关者进行沟通。

### 3.7.1 质量成本

有些技术可以用来量化测试的大部分量值。但是，高级大纲——也就是本书——侧重于用质量成本来度量量值和测试效率。质量成本有时候也叫做劣质成本，用来提醒人们劣质费钱而优质省钱。<sup>17</sup>

从核心上看，质量成本是一个会计学技术，它将项目或者运营成本分成4个类型。

- 第一类为预防成本。预防成本就是我们用来预防缺陷发生的费用，培训开发人员的费用也属于预防成本。

---

<sup>17</sup> 在本书中，和高级大纲一样，我们只讲质量成本。但是，如果您想要学习更多关于度量测试的其他量值，推荐您读一下我的“Critical Testing Processes”一书。



- 第二类为检测成本。检测成本是指用来发现缺陷的成本，以及即使没有检测到任何缺陷也必须花费的成本。花费在测试计划、分析、设计和实现的成本是检测成本，同时，某些测试执行的成本（执行通过的测试）以及测试结束活动的成本也是检测成本。
- 第三类为内部失效成本。内部失效成本是指我们发现缺陷后的费用。测试执行的额外费用，特别是重新测试，针对修复的确认测试，安装第二次以及所有后续的测试发布版本等诸如此类的成本，都是内部失效成本。程序员修复缺陷，以及和其他项目支持人员，比如说版本发布人员，创建测试发布相关的费用也都是内部失效成本。
- 第四类为外部失效成本。外部失效成本是指由于在版本发布前没有发现和移除缺陷而引起的费用。一半以上的技术支持成本或者服务费用是由于外部失效引起的。程序员花费时间修复现场故障是外部失效成本。在某些情况下，外部失效常伴随着实际的处罚或者业务的丢失。例如，有些公司和他们的顾客有相关服务协议，对于失效有相应的处罚。有些公司不得不担心潜在的顾客选择不和他们合作，比如说当顾客的某个电子商务网站停止运营时。

当 RBCS 执行测试评估时，我们一般会通过质量成本来度量测试（如找到缺陷）的效率。从一个顾问的角度看，在每个实例中，检测成本和内部失效成本比外部失效成本要少得多。这种情况下测试就变得非常有价值。我找到一个例子，在该实例中，测试节省了外部失效引起的潜在的公司费用，因为预发布检测发现了缺陷并将其移除从而减少了费用。

表 3-18 使用本书中多次引用的因特网相关实例来进行质量成本分析。我们会从左上角开始然后慢慢往下看如何执行这个计算。

表 3-18 实例：质量成本投资回报率 ROI

检测成本		外部失效成本	
测试预算	\$1 000 000	维护花费	\$3 000 000
资产未来价值	100 000	缺陷相关百分比	50%
再测试成本	500 000		
净检测成本	\$400 000	净外部失效成本	\$1 500 000
必须修复测试缺陷	1 500	必须修复的发布之后的缺陷	500
平均检测成本	267	每个缺陷外部失效成本	3 000
内部失效成本		投资回报	
测试缺陷修复成本	750 000	必须修复测试的缺陷	1 500
重复测试成本	500 000	每个缺陷节省金额	\$1 900
净内部失效成本	\$1 250 000	测试净收益	\$2 850 000
必须修复的测试缺陷	1 500	净检测成本	400 000
每个缺陷内部失效成本	833	测试投资回报	713%

首先我计算了检测成本。正如我所提到的，测试预算的一部分形成了检测成本。所



以我以\$1 000 000 作为测试预算。接着，我扣除了为后续项目创建和获得设备价值所花费的时间数（基本上是测试结束的成本）。最后，我扣除了重复测试的成本——确认测试缺陷修复和受缺陷修复影响的回归测试所花费的成本——因为这些是内部失效成本。还剩下\$400 000。这里有 1500 个必须解决的测试缺陷，也就意味着我们的平均检测成本是\$267。

接着，我计算了内部失效的成本。这个包括测试预算的余额，重复测试成本，你可以看到在这里我都加上了。这里也有开发人员用来修复缺陷的工作量。我是基于 3 个月的系统测试进行估计，而且假定在这 3 个月内产品的特性都已经完成。净内部失效成本是\$1 250 000，平均内部失效成本是\$833。因此，每个缺陷的平均预期发布成本——包括检测和内部失效——是\$1100。如果一个缺陷在发布之后被发现的成本超过\$1100，测试就是节省了公司的金钱。

让我们来看一下这张表的右下角，看我们是否节省了钱，以及节省了多少。我一开始计算了外部失效成本，维护成本——包括开发人员、测试人员和技术支持——在发布的前 6 个月是\$3 000 000。我估计大约 50%的花费是由于缺陷（我认为实际的百分比要更高一些，但是就像我之前说的一样，我们想要让这些商业用例数据保守一些。）下一个外部失效成本是\$1 500 000，也就是在发布后发现了 500 个缺陷。这个也给了我们一个发布后缺陷的平均成本是\$3000，也就是每个缺陷假如在测试中发现，可以节省\$1900。

最后，我们来计算一下我们的投资回报率。因为每发现一个缺陷就节省了\$1900，总共发现了 1500 个缺陷，测试总共节省了\$2 850 000。从百分比角度来看，我们在这个项目上的测试成本——如果我们减去结束成本和缺陷相关重复测试成本——即检测成本\$400 000。所以，如果我们用\$2 850 000——测试的净收益——除以\$400 000，就是 730%。还不错的成绩。

### 3.7.2 测试的其他价值

由于高级大纲侧重于质量成本，我们只是简单地介绍一下和测试相关的其他定量和定性价值。

根据一些更深入的分析，我从测试因特网应用的项目中发现下列额外量值：

- 由于已知缺陷（经常和测试识别出的补救办法相关）而减少了支持呼叫时间，从而节省了\$150 000。
- 已经通过的测试降低了失效风险。基于现场失效的成本和测试首次运行时测试失效频率和可能性，可以得出节省了\$250 000。
- 降低了因项目跟踪不良导致项目失败的风险，从而得出提供的信息价值为\$200 000。<sup>18</sup>

这个项目也表现出了一些额外的定性价值。首先，公司可以对发布有些信心，特别是对我们所管理的测试领域进行了成功的缺陷清除，比如性能和可靠性方面的缺陷（尽

---

18 我在 RBCS 公司网站上的一篇文章中描述了这些值和它们背后的分析，“IT 经理应该知道测试中投资回报率的哪些内容”。Capers Jones 在“Estimating Software Costs”一书中描述了项目失败的风险是由于项目跟踪不良。

管仍留有相当数量的缺陷，你可以看到，我们的缺陷检测率只有 75%)。其次，没有出现因为更新软件失败引起的召回，只出现很少的无用附件的退回。这在某种程度上要归功于我们对硬件和软件更新过程的全面测试。

### 3.7.3 测试商业价值的练习

考虑一下在 HELLOCARMS 项目第二次迭代的结尾（如前一个练习中所示）。列出在该时刻由测试交付的定量和定性价值。

如果你在教室里，那么最好分成 3~5 个小组。如果您只有一个人，需要自己做这个练习。每个小组完成之后，讨论一下结果。

建议用 15 分钟来完成这个练习，包括讨论时间。

### 3.7.4 测试商业价值练习答案

我们来看一下在课程中提到的适用的测试价值。

定量价值

- 找到必须修复的缺陷：这个在之前练习的图表中很明显地显示出来。
- 找出可以推迟解决的缺陷：我们并没有对推迟解决的缺陷进行细分，但是我们可以假设某些缺陷推迟是由于缺陷解决率低。
- 通过运行测试减少风险：这个在之前练习的图表中也很明显地显示出来了。发现大量的测试失败，尽管它并不是一个令人喜欢的结果。
- 传递信息：在之前的练习中，我们可以基于那 4 张图表肯定地得出测试结果不支持部署。如果这个信息被合适地传递到项目管理团队并被其接受，Globobank 公司就有机会在发布时避免尴尬和过多的花费。

定性价值

- 改进质量声望：再说一次，如果项目管理受之前练习中的发现影响决定在第二次迭代结束阶段不进行部署，那么这里我们肯定提供了价值。
- 平稳发布：我们肯定尝试过理顺发布过程，基于第二次迭代的改进我们将继续改进并做得更好。但是，我们要对在第二次迭代后期的缺陷高发现率和测试高失败率的原因进行深入调查，以增加我们在这里要提供的价值。
- 增加自信：我们没有检查 HELLOCARMS 测试的覆盖率标准，所以我们不能说在这里是否做了充分的工作。
- 保护法律责任：由于测试确实覆盖了安全和监管相关的特性——对银行来说这是两个最主要潜在的法律风险——在这里我们有附加值。
- 减少任务和生命损耗：尽管这不是一个安全关键应用，但它是一个业务关键应用。不能进行房屋净值贷款、房屋净值信用贷款以及反向住房按揭贷款的银行就丢失了最重要的获利手段。

你也可以找到其他适用于本项目的定性或者定量的价值。



## 3.8 分布式、外包、内包测试

### 学习目标:

(K2) 列出 3 种测试策略（分布式、外包和内包）的风险，共同点和区别。

不久前，软件和系统项目通常遵循内部开发的方法。单个团队负责实施大多数项目。测试团队是同一个公司的一个部分，他们都在同一栋办公楼办公。现在，很多情况下一个公司可能将测试工作分散到不同的团队中。公司开发软件时，可能只会雇佣其中的一个或者两个团队。测试、开发和其他工作可能在 2 个、3 个、5 个，或者更多不同地方的团队来完成。这到底是怎么一回事？

其实，在这个过程中，不止发生一件事，极有可能发生一件、两件，或者 3 件不同的事情。我们来看一下这些不同的情况：

- 如果测试工作在多个不同的地方完成，那么测试工作就是分布式的。例如，假设你的公司在班加罗尔有一个测试团队，在布拉格有另外一个测试团队，而这两个测试团队共同对一个项目进行测试。这种分布式的方式会带来很多新的管理问题，比如说工作中的合作。
- 如果测试人员和项目团队中的其他人不是同事，而且不在同一个项目中，而是在一个或者更多的地方执行软件工作，那么这个测试工作就是被外包的。如果超过一个地址，那就是分布式外包。比如，如果你雇佣 RBCS 在新德里的测试团队为你做所有的测试，那么你就是外包了你的测试。但是，如果你保持一部分的测试在你自己的团队中，然后将剩下的测试外包给新德里的团队，那么就是分布式的外包测试。外包也会引起很多新的管理问题，某些问题因为不同团队的距离（包括地理距离和时区）和涉及团队中成员的数量增加而加重。
- 如果测试人员和项目团队的其他人员在同一个地方工作并执行测试工作，但是这些人员和项目团队的其他人员不是同事，那么这个测试工作就是内包。比如说，你雇佣 RBCS 的一个测试员来为你的项目测试工作，那么你就内包了你的某些测试。内包也会引起一些新的管理问题，比如说要确保带到团队中的人员带来了你需要的技能。

我们可以将相同的分类方法用来对开发和其他项目功能进行分类。这种通过这 3 种分类方法将测试、开发、项目管理、发布工程，以及其他项目职能进行潜在联合非常重要——但是它们也可以提供一个巨大的迷雾，在这种情况下，如果这个组织不够仔细的话，有过错的一方可以隐藏其应承担的义务。

你可能注意到我将离岸、近岸等名词放在一边了。我认为这些名词被市场人士用来区别它们的服务。从这种程度上，这些名词有不同的意义，这个意义是相对的——比如说，离哪一方的岸？近到哪一方？——因此通常是不容易处理的。

在这一部分，我们来看一下它们是如何影响测试的，特别是测试管理。这些影响，有的是共同的，而有的是特殊的。我们先从适用于分布式、外包和内包测试的常见问题

说起吧！

我们想要保证整个项目团队是朝着正确的方向运转，那么定义良好的期望是非常重要的。在所有我们需要考虑的案例中，有两个或者两个以上的小组参与进来。如果这些小组认为他们在做——或者说试图做不同的事情，那么你就不要为部分或者完全的失败的到来感到奇怪了。

如果项目中每件事情都进行得很完美，那么我们可以将工作分发到全球去做，到最后再将这些完工的工作产品聚集起来。但是事实上，人们需要在项目过程中讨论、沟通、合作、学习甚至是改变项目方向，所以清晰的沟通渠道是非常必要的。这种沟通的渠道不可以很随意，不可以像公司同事喝咖啡、抽烟、健身的时候那么随意。

作为测试经理，你需要仔细协调发生的事情，而这在内部项目中你可以很自然地运用最小的工作量即可达到目的。你必须仔细地计划和管理相关的测试后勤工作，比如说相关的硬件、软件、工具许可证、信息流、测试数据入口，以及其他的事项。你还需要管理存在的不同地点、时区、文化和语言的差别。地点和时区的差异是非常明显的，所以不要低估了这两个因素。

第一个就是出差到远程地点解决问题的效果。我在另一个半球的城市通过 1 个小时的面对面会议，成功地解决了一个棘手的测试沟通和合作的问题。这个问题在我最后说“我要去现场”，坐上飞机走之前，已经困扰了我们项目好几个星期了。

第二个问题就是组合管理不同时区的困难。是的，在至少存在两个不同的时区的情况下，你就可能发现一个相互合适，或者可能相互不合适的时间来进行电话会议、视频会议、在线聊天、在线研讨会，或者其他团队信息渠道。如果这个时间对大家来说都很方便，那么你作为一个经理让项目成员参加这些常务会议就不会有很多麻烦。试着将一个常务会议放在早上 7 点，每周 1 次，然后看一下这些员工（比如我）会怎么回应。现在，试着找一个合适的时间在以下城市进行电话会议：澳大利亚悉尼、得克萨斯的奥斯丁和德国的柏林。或者是伦敦、东京和圣迭戈？或者这 6 个一起？这些时区中的夏令时都是什么时候开始和结束呢？

文化和语言的不同其实是更加明显，这让我们感到吃惊。是的，文化在一定程度上是指像食物、宗教、节日、禁忌、喜好等一些东西。这些东西从全球来讲表现出惊人的不同而且给你提供了无穷尽的机会让你在社会和商业情景中来做和说一些完全不合适的事情。在你开始和项目中的一群人开始工作之前读一本关于文化和历史的书是非常有用的，而且可以帮助你处理由此引起的麻烦。

但是文化同样也包括商业文化，即使在同一个国家中也有很大的不同。比如说，在东亚地区，商业交易一般都是有礼貌而且不会有冲突的。商业人士经常表现出高度的合规性。当出现了一个问题，团队之间趋向于间接和私下地工作来解决问题。因此问题解决得很慢。但是，我有一次坐在东京的一个会议上，交付我们对客户测试情况的评估，当时实在是被我们的客户对我的商业伙伴的态度惊呆了，他们对我的商业伙伴的每个观点和主张都进行了质疑。我在以色列做了很多业务，那里的人在进行直接和坦率的讨论这一点上有着很好的声望，从没有在与这些喋喋不休而且直言不讳的以色列人的讨论中发生过我处理不了的事故。



如果大家不能达成共识直接解决问题，那么事情的进展肯定趋于缓慢。这在通常的情况下是会发生的。RBCS 曾经参与一个在东亚的公司的一个大型的课件开发和交付系统，这个交易从开始讨论到签署合同只花费了几周很短的时间。但是和美国公司的一个类似交易持续了几个月才完成。所以特别小心对文化存有思维定式。在这个全球化和多样化的世界，墨守成规会让你误入歧途的。

关于语言的问题，这里有两个需要你记住的很细小的、但是有着难以言喻的联系和相互作用的真理。首先，理解一门语言不是一个简单的二元状态可以表达的。比如说，我可以读西班牙文。如果我有随身字典我的西班牙文也可以写得很好。但是我在说和听西班牙语上面有困难。所以，如果有人对你说，“哦，是的，我们所有的测试人员都能够理解英语”，请不要想当然地假设在该项目中就不会有语言问题。

其次，当你用一种语言来理解和表达你自身的时候——任何语言，包括你从小到大一直说的——你所有的观点都不可避免地会通过一个由复杂的过滤器进行过滤，这个过滤器包含了你过去的经验、文化、你的偏好、个人情感等因素。当你尝试和别人交流时，你和他分享的这些因素越少，那么你们之间沟通交流就越可能出错。

另一个问题是在各小组间采用的方法要取得共识。缺陷怎么报告？需求怎么收集和写入文档？进行哪几种检查？比如说，如果你认为 ISTQB 基本测试过程中的所有活动以及每个活动中的所有任务，两个小组可以执行任务中的大多数，但是每个小组可以用稍微或者完全不同的方法去执行所有任务。那么你首先要确认这些方法的不同之处，最理想的情况是在测试计划阶段，而不是测试执行阶段。你一定要决定哪些方法会影响你的测试，而且只有这样你才能采取行动解决这一问题。

同样的，要确定管理好生命周期潜在的问题。如果一个小组使用敏捷方法论而另一个使用了顺序方法论，这是一个明显的不一致。两个小组可以遵循相同的生命周期，只是不要对在不同的小组中引起的相关问题有什么想法。

最后，万一这个额外讨论没有让你看到事情的真相。要记住，和内部开发的项目相比，你将不得不处理更多不同的项目风险。务必保证计划和管理这些增加的项目风险。

### 3.8.1 特殊的分布式、外包和内包测试问题

哪些问题和分布式测试工作特别有联系？最重要的一个是将测试工作分配到不同的地点。你必须明确地进行这个划分而且要划分得详细。你不能依赖某人所给予的关于哪种测试需要在哪里执行的模糊声明。你必须明智的进行这个划分。一定要考虑到不同小组中不同的技能和资质，以及将这些工作分配到合适的地方。

在测试策划以及整个测试工作中，仔细检查不同小组之间的间隙和重叠的部分。间隙的产生就是每一个小组和经理都认为某些测试条件在某些地方被覆盖了但是实际上没有人去做。间隙就会导致在测试工作中产生错误的信心，在测试执行中会有潜在的不愉快的结果，而且通常在交付时会增加剩余的质量风险。

重叠就是两个或者多个小组以及他们的经理认为某些测试条件被他们的小组独自覆盖，而实际上是所有的小组都覆盖了该测试条件。重叠会导致低效率，在测试执行中会导致潜在的重叠工作，如果同个测试的结果出现不一致，就可能会导致对测试结果解析

的困惑。

最后,采用单一统一测试汇报格式。所有的测试结果应当能做成某些单一集合的报告。产生这些报告也不应当是费力的任务。理想的是,每个人将他们的信息放入一个中央测试库并从这个库中自动生成所有的报告。

讲完了分布式测试的问题之后,我们来看一下与外包内包测试工作相关的问题。首先,我先来讲一下引入这些问题的一点历史。在20世纪80年代和90年代的时候,计算机硬件业务经历了一个强大的趋势,将笔记本电脑、台式电脑、服务器、硬盘、光驱,以及其他的部件的开发和生产外包给了远程的承包商,特别是在东亚的伙伴。在20世纪90年代和2000年代的时候,软件业务开始顺应强大的趋势开始遵循相同的路径,这一次有一个更加全球化的工作分布,不仅仅包括东亚,也包括南亚、东南亚、非洲、南美、东欧和欧亚大陆。

在这两个案例中,硬件和随后的软件,其主要的驱动力来自节约成本。对低成本外包承包商和当地开发人员之间可能的质量间隙的认知驱动了很多承包商追求像ISO 9000(在硬件领域)或者SPICE/CMM(在软件领域)这样的认证。但是,这些认证主要服务于市场的机制,用来消除潜在的销售上的问题。即使承包商真正完全实现了CMM或者SPICE,从测试的观点来看,这并不一定能提供必要的帮助,后面我将对这个问题进行深入讲解。

同样的,承包商打着比如“正确地选择外包”以及“追逐太阳”的口号来消除另外两个可能的对销售的歧义。“正确地选择外包”阐述了存在于迅速成长的高技术经济所谓的技能问题,特别是从资深人士与初级人士的比例来看。例如,有时候我听到新兴市场上的测试人员将他们自己称为“资深人士”,因为他们有两到三年的工作经验,做了三到四个项目。在大多数老道的高技术经济中,“资历深”最起码要有10年工作经验。如果是管理经验,那肯定还要更多的年限。“追逐太阳”表明了在很多案例和项目风险存在的非常现实而且很成问题的时区难题,比如试图指挥和管理发生在几千公里外,5个或者10个时区之外的工作。

我曾经参与整个外包的过程。我曾经工作于基于美国的外包测试实验室,我帮助客户使用新兴市场外包测试实验室,RBCS提供了覆盖全球的内包和外包的测试服务,这个模式是行得通的。它可以帮你节省很多钱,而且你还可以管理这个风险,但是你还是需要对这些市场策略和销售口号保持清醒。

首先,确保你已经为成功组织好了。这个也就是指我之前提到的对方法和生命周期取得共识,还包括沟通渠道和后勤的组织,但是它也涉及合同以及相关的理解。如果你期望能够检查一个开发承包商的测试结果,你最好确保这一点是在合同中定义的。要不然,当你要求他们的测试数据、测试用例、测试工具、缺陷报告和测试用例跟踪信息时会大吃一惊。

其次,仔细考虑你是否需要一个独立的测试伙伴,如果是,那么请选择一个伙伴。当然,因为RBCS提供外包测试服务,我们相信通过独立的测试伙伴来帮助你的外包项目是一个非常明智的方法。要不然,不管你的开发承包商如何组织他们的结构,你都会遭遇测试独立性的问题,他们会对所有的测试结果进行自我修编和管理修编。



第三，不管是外包还是内包，保证你的主要人员和他们的主要人员之间有充分的人员交流。这对外包来说当然比较困难，因为你不可能经常见到他们。但是，内包测试人员被孤立的现象也会经常发生。

正如之前所说，你需要适应文化。在外包和内包的案例中，一个额外的文化问题就是企业文化问题。注意到我说“适应”而不是“改变”。如果你雇佣了外包或者内包承包商，千万不要天真地期望他们会改变他们的文化以适应你的。同时适应两边的文化是最好的结果。

最后，确保你将注意力一直集中在完成工作这一需求上。即使每个人都属于同一家公司，分布式工作中也会出现很多琐碎的争吵，一旦有外包和内包公司的加入，情况会变得更加严重。当人们认为外包或者内包公司对他们的工作岗位构成威胁时，这一点就更加成立。作为一个经理，不要为办公室有烟雾或者其他类似转移注意力的事情而浪费精力。而是将注意力保持在测试工作的完成上面。

保证你的外包或者内包伙伴也持有相同的关注焦点。服务公司的一个不太好的花招就是在项目中提供一个“高级顾问”或者类似的人员，可能是在现场的，这个人花费时间来找到增加你的预算的途径。而且，你还要为他所做的事情买单。

注意这样的花招，并且不要容忍它们。是的，服务公司在你有需要的时候应当提出额外的服务。要求他们这样做，从而可以帮助你，因为他们可能可以解决别的问题，包括你还不知道的其他问题。但是，如果你请的测试顾问为你的项目工作，那么这个人应当为你的项目工作，而不是在寻找其他新项目的同时还要向你收钱。

现在，可能听起来有点伤感或者好心肠，但是实事求是地讲，分布式、外包和内包测试需要彼此之间的信任。分布式、外包和内包测试都是通过相互之间的合作来完成工作的。信任很关键，因为在和某人合作的同时还要搜集证据来反对他是很困难的。

鉴于我提到的所有的问题，包括相关的公司、文化、语言和地理边界，对这些方法保持怀疑的态度是很简单的。但是，我提这些事项是为了帮助你管理这些事项，而不是为了提供反对使用这些技术的论据。如果你仔细地管理分布式、外包和内包测试，每个测试团队都可以合理地执行他们的职责。

基于这点，对任何事情的持续怀疑和缺乏信任会导致效率低下和延期。在相互之间不信任的情景下，我看到很多人花费大量的时间，用来验证已经发生的行为，对发生的小事每次都兴师问罪，玩弄公司政治。

如果合理地计划和管理分布式、外包或者内包测试，你可以很自信这个团队在做正确的事情。在这种情况下，彼此信任可以帮助每个人维持正确的焦点。

一定要得到信任而且双方都要得到。当然我不是说盲目和天真地信任承包商，那很可能被骗。我想要说的是，如果你只是期望平常的期望，使用和其相关的清晰的性能度量，你应当可以决定每个人在合理地做他们的工作。如果有人不是，那么你要使用清晰的一组数字来揭示性能问题以及指导改进的方法。

### 3.8.2 能力成熟度模型集成（CMM）和测试

前面曾经提到过会回到关于 CMM 和 SPICE 的问题以及它们如何与测试相关联。

表 3-19 展示了 12 个主要的测试过程。在这个表中，我对 12 个关键过程的每一个进行了 CMM 评估。斜体的数字指的是 CMM 的级别，其中引入了某些测试过程的某些元素。注意我不是说所有的元素都是存在的，或者所有引入的元素都做得很好，但是至少在该 CMM 级别测试过程的某些事必须完成。<sup>19</sup>

表 3-19 CMM 和测试

关键测试过程	CMM	关键测试过程	CMM
测试（计划、准备、执行、完美）	2 3 4 5	测试系统设计和实现	3
定制环境测试	3	测试发布	2 3
质量风险分析	3 4	测试执行	3
测试估算	2 3 4	缺陷报告	3 4 5
测试策划	2 3	测试结果报告	3 4
测试团队职员，技能	3 5	变更管理	2 3 4 5

粗体数字指出了 CMM 级别，你可以假设对该测试过程进行合适的处理。换句话说，所有的关键元素就在那儿，而且假定某人已经按照获得的 CMM 级别在实践，那么你可以假定他们的这些关键过程做得很好。

SPICE 在方法和哲学方面类似于 CMM，可以合适地在分析中展示类似的相同的结果。某些人断定 CMMi 改变了这个形势。我不认为如此，CMM、CMMi 和 SPICE 没有覆盖测试领域的原因是，从哲学上讲，他们都不认为测试是通往更好的质量的主要途径。我们会在第 8 章更仔细地讨论这个问题。

所以，要记住的重要的事情是，CMM、CMMi 和 SPICE 可以帮助公司建立质量的基础，但它们不会保证全面的测试。事实上，就像你会在第 8 章所见的，一个公司宣称通过了 CMM，从测试角度来说，对该公司的测试应当更加仔细。

### 3.8.3 分布式测试的案例研究

让我们用分布式和外包测试的两个例子来结束这个部分。第一个例子，一个成功的分布式测试，是该课程中一直使用的因特网相关项目的案例。

在这个项目中，供应商的测试团队做了大多数的硬件测试。我们将一些最重要的关键测试由第三方硬件测试实验室重新做了。我不记得有任何供应商测试不充分的情形。在接下来的软件测试中，当他们的成品硬件集成到我们的整体系统时，我们在成品硬件单元中没有发现不寻常的硬件问题，所以我们有了一个完全外包和双倍分布式的硬件测试工作。我们可以说这个供应商的测试团队其分布式测试是成功的。

对于第二个例子，一个不成功的分布式测试，考虑这个网络相关项目。我的客户从一个供应商那里买了一个邮件服务器组件，尽管只是买了一个硬件。强调一下，这个供应商在交付前测试了他们的组件——至少他们说做了。但是，当我们试着将他们的软件

<sup>19</sup> 12 个关键测试过程来自我的“Critical Testing Processes”一书。我使用自己的测试过程框架，但是你用特定的测试过程框架来进行这样的分析并不会改变结果太多。试着用 TMM、TPI、STEP、T-MAP，这样你就可以理解我所说的意思。



组件集成到我们的系统，和我们的硬件经验不一样的是，事情没有很好地进行。有很多严重的邮件相关的问题被发现了，包括功能、性能和可靠性方面的问题。

所以，这个供应商测试团队的分布式测试失败了。相同的项目，相同的负责人（也就是，我和我的团队）为什么会发生这种事情？

一个重要的区别是我们可以约束和管理硬件测试。我在台湾地区待过一段时间而且和硬件测试经理直接工作，他非常诚实而且对我非常坦率。相反，邮件服务器承包商掩盖了他们的测试结果，拒绝交付一个他们已经答应交付的测试工具，一旦要求他们交付，就坚持要我们使用他们的一个初级测试人员作为“顾问”（大约\$1 500/天）。

你可以说我的客户很幸运地选择了硬件供应商，但是选择了一个不好的软件供应商。但是，我要说的是其中根本的区别并不是幸运不幸运的问题。我的客户联系选择硬件供应商，我认识这个硬件供应商和他们的测试团队，而且每个人都知道我们想要参与他们的测试。这个硬件供应商没有任何事情可以隐瞒。但是，另一个经理选择了软件供应商。我之前没有和这个经理及这个软件供应商共事过。我不相信这个经理事先对承包商设定了恰当的期望。所以，就像所有关乎测试成功失败的因素一样，现实的达成恰当共识的期望也很重要。

### 3.9 测试管理问题

#### 学习目标:

（K2）比较探索测试、综合系统测试和安全关键系统测试的管理问题，包括策略、优缺点、充分性，以及对策划、覆盖、监视和控制的影响等话题。

我们将会在本章讨论一些测试管理方面的问题，这些也许是你作为高级测试经理必须管理的内容。这些问题主要有4个方面：

- 管理反应式测试策略和基于经验的测试技术。
- 管理综合系统测试。
- 管理安全关键系统测试。
- 管理非功能性测试。

我们会从管理反应式测试策略和基于经验的测试技术这个问题讲起。

这几年对于反应式测试策略的抱怨主要集中在3个方面，例如缺陷检测、软件攻击，以及探索性测试，成为管理的难题，对测试没有文档化记录，就不易将其扩展到更大范围的测试团队。有些探索性测试的支持者对这些批评反应非常强烈，而且他们中的某些人已经尝试发明一些技术来管理、来进行积极地回应。

基于会话的测试管理是解决其中两个问题的方法。它被设计用来管理探索性测试，但是也可以将它应用到任何基于经验的技术，以及所有反应式测试策略。

在基于会话的测试管理中，将测试执行工作分解为测试会话。一个测试会话就是一个基本的测试工作单元。通常来说有时间上的限制，比如30~120分钟。被分配到一个测试会话的时间叫做时间箱。如果你还记得我们之前关于获得价值的讨论，我就说在测

试执行中，我们可以将测试用例作为测试工作单元对待。在这里概念是一样的。

对于测试人员来说，测试会话应当是测试执行不被打断的一段时间。测试会话应当集中于具体的测试对象。他们应当集中在特定的测试目标，这个目标可以写在测试示意图中的。你可以认为测试示意图是非常概要的测试用例，或者只是一到三句话。

每个会话都应该输出一个报告，也就是获得测试人员测试的细节。报告可以和手写的测试脚本拥有同样的详细程度，不同的是该报告是在测试执行后创建的。这个方法和基于分析的测试策略不同。基于分析的测试策略，其测试设计和实现发生在测试执行之前，而且大多数文档是在测试设计和实现阶段输出的。

让我们来回顾一下在早期的图 3-39 中所显示的测试用例概要工作表，这个表在标注计划的小时数的同时也列出了测试用例。按照在每个测试用例中所提供的细节级别，你可以很容易地使用该工作表来管理和追踪会话。其中的差别是，正如我之前指出的一样，一个分析方法中，测试人员会在测试设计和实现中创建大多数的信息，但是在反应式测试策略方法中测试人员会在测试执行的时候，捕获会话报告的信息。

现在，基于会话的测试管理给测试提供了一个可管理的过程，我们之后会再深入地讨论。我们现在也有了一个收集和汇报测试结果的方法——这也是我们经常用的，因为至少从 1990 年开始像我这样用测试用例来跟踪工作已经成为测试的一种惯例。<sup>20</sup>但是，这种方法还是没有得到规模应用。我们待会再接着讨论这个问题。

首先，思考一下什么是过程？在测试会话中测试人员应当做什么？该技术的支持者阐述了它的 3 个主要阶段：

1. 会话建立。包括建立测试环境以及保证测试人员理解被测产品或系统的范围。从某种程度上说，这个阶段和预防性测试策略下运行有脚本的测试时创建前置条件这个活动相对应。但是，它也包括某些测试环境的建立任务，这个任务在预防性测试策略，像基于风险的分析测试下，是在测试执行前进行的。

2. 测试设计和执行。由运行针对测试对象的测试以及寻找问题组成。如果你使用探索性测试，这个阶段就是探索性测试拥护者所说的“同时学习产品，运行测试，以及决定随后要做的测试”活动。但是，你可以在这种方法下使用维塔克尔（Whittaker）的软件攻击方法。<sup>21</sup>注意，其中包含了设计活动。再一次强调，在类似基于风险分析测试的预防性测试策略中，测试设计会在测试执行前发生而且不会形成影响发布的瓶颈。

3. 缺陷调查和报告。基于脚本的测试发现问题的时候，也会触发这个过程。但是，由于我们没有一个详细的书面测试用例，开发人员以此作为参照就没有什么意义。所以，比起基于预防性测试策略的缺陷报告进程，测试人员在反应式测试策略中使用更加谨慎和详细的缺陷汇报过程；预防性测试策略通常会产生可参照的测试脚本和测试数据。

当我描述这些活动的时候可能听起来像是连续的，在大多数情况下这些活动是重叠的。另外，根据汇报会话的结果（马上就会讲到），对于给定的测试示意图我们可能重复

---

20 我在 1999 年出版的“Managing the Testing Process”一书的第一版中讨论了这个问题，但是就算在那时，使用这种跟踪电子表格也已经成为了一种惯例。

21 Whittaker 在他的“How to Break Software and How to Break Software Security”一书中描述了这些攻击技术。我在高级测试分析的配套卷中会涉及这些话题。



引用这个过程。

现在，当缺陷报告更接近一个反应式测试策略——如果有更多详细的内容来补偿对所记录的测试用例和测试数据的缺失——这个汇报结果就会完全不同。至少在使用基于会话的测试管理时会是这样的。在使用脚本测试的预防性策略时，测试人员会对测试脚本的偏差或者扩展做上一些标记，捕获测试状态、使用的环境，以及其他比如测试用例概要工作表需要的数据，并将它们移到下一个测试。

但是，在基于会话的测试管理中，因为书面的测试脚本不存在，测试人员必须获得被测试的信息用来填充文档的缺失。这些就叫做会话报告或者会话表。在会话报告中会有什么内容？会话报告内容按照项目所需要的文档需求会跨度很大。但是，基于会话测试管理的倡导者建议以下的领域：

- 会话示意图。正如我之前所说的，这是一个用一到三句话来描述需要测试的内容。
- 测试人员姓名。
- 开始日期和时间，以及可能结束日期。
- 任务分解。我们在测试过程中做什么？注意到，如果我们需要一个测试的非常详细的描述，我们会发现我们处于这样的一个环境，我们将有效地创造大量的测试脚本分布在发布的关键路径上。而不是像预防性测试策略那样将测试设计和实现、测试设计和实现平行进行，从而偏离关键发布路径。
- 度量，例如用来建立、测试和缺陷汇报的时间（也就是应该分配在时间箱中总的的时间）。
- 数据文件。包括用来执行测试以及由测试生成的数据文件。在某些项目中，你可能在飞行途中创建充足的数据作为一个 30~120 分钟测试会话的一部分，但是在更多的项目中这些数据是不够的。重申一下，当测试实现的一部分在测试执行前执行时，预防性测试策略会使得这个工作不会成为影响发布的瓶颈。
- 测试注释。这个可以是捕获的关于测试会话的任意形式的信息。
- 问题。问题包括出现的担忧，例如硬件缺失导致不能完整执行测试示意图。但是，它也包括在会话中发现的额外的测试机会，可能会导致新的测试示意图的创建。
- 缺陷。这个可能会列出从缺陷跟踪系统中导出来的缺陷序列号，但是它可能也会包括从每个缺陷报告中得到的概要标题。

每个测试会话以测试人员和测试经理之间一对一的报告会作为结束。这些报告会在需要的时候，可以在测试执行中开展。

在汇报过程中，测试经理检查会话报告，要求测试人员澄清问题，并和测试人员讨论他们所发现的问题。根据讨论的结果，测试人员和测试经理可能改进潜在的测试示意图或者某些其他的测试示意图，从而激发新的会话。测试经理可以基于测试人员的反馈调整对剩余测试工作的计划，尽管可能大多数这些调整都是很细微的。测试经理也可以估算和计划将来的会话，试着管理在测试执行阶段的剩余时间。

现在，这些会话日程是可变的。一些对于基于会话测试管理的支持者使用一个缩写 PROOF 来进行说明：

- Past（过去）：在会话中发生了什么？你过去做了什么以及你看到了什么？

- Results (结果): 在这个会话中得到了什么? 发现了什么缺陷? 哪些是可行的?
- Outlook (展望): 还有哪些需要做的? 哪些后续的测试会话我们应该尝试?
- Obstacles (障碍): 什么成就了一个好的测试? 哪些是你想做但是做不了的?
- Feelings (感觉): 你觉得这个测试会话怎么样?

我在测试汇报会上使用的一个更加简单的日程表:

- 哪些是可行的?
- 哪些是不可行的?
- 哪些缺陷你发现了(通常这样会要求进行缺陷报告的评审)?
- 哪些有趣的事情发生了?

我在之前提到的基于会话的测试管理, 虽然解决了反应式测试策略中某些过程和文档的问题, 但这只是一个很不全面的解决方案。其中一个不能解决的问题就是规模的缩放。

通过扩展测试管理技术, 我指的是可以经得起很多情况考验的适用于大多数项目的技术。这就必然包含了我们在接下来的段落中要讨论的关键属性, 这在基于会话的测试管理中是缺乏的。

测试管理技术要既耐用又通用, 就应当能够控制任何地方的 1~100 人规模的测试团队。根据需要进行测试人员和测试经理之间的报告会在这里就限制了规模的扩展性。我通过举行一天开始或者结束时的小组报告会解决了这个问题。

测试管理技术要既耐用又通用, 就应当可以用来处理在不同时间上班的测试团队, 可能是在不同的地点和时区。再一次强调, 根据需要进行测试人员和测试经理之间的报告会限制了这里的可扩展性。如果报告会的时间选择正确的话, 我发现这个在一天开始或者结束时举行的小组报告会是一个解决分布式时间的办法。

测试管理技术要既耐用又通用, 就应当可以适应在测试团队中语言的多样性。按需进行报告, 在与只使用单一语言的团队进行电话会议时困难就很大, 想象一下如果还有语言不同的情况会如何。我发现我可以组织好通晓多国语言的测试团队的报告会, 通过电话会议或者网络会议, 通过增加的文档, 包括提供足够详细的测试示意图或者测试用例或者测试脚本——不管我们怎么称呼它们——以保证在测试会话开始之前对整个团队就有清晰的认识。

测试管理技术要既耐用又通用, 就应当尽可能经济地雇佣测试团队, 团队中可以有合同工, 外包、内包测试人员。很明显, 所有基于经验的测试技术自然依赖于测试人员的经验和技术。在测试合同工的人才资源库里不会总是有熟练的测试人员——至少不是以这个项目所能承受的价格。如果选择了正确的测试顾问, 那情况就不错。但是, 如果你不得不与你可以获得的合同工, 在你的外包商组织中工作的测试人员, 或者基于最低竞价原则中标的现场测试团队合作, 这个技术就会置你于危地。通过在测试示意图中加入更多的细节——将它们转换成测试脚本, 尽管只到需要的详细程度——为你能获得的测试人力资源提前做好使用的准备, 就可以解决这些问题。

最后, 测试管理技术要既耐用又通用, 就应当在测试执行前进行工作量和时间的估算, 最好是在测试计划过程中。探索性测试的支持者宣布已经解决了这个问题: 只要找



出所需要的会话数，估计每个会话的时间箱，做一点点的算术，看，这就是你的估算。注意，尽管，这是一个循环的论证。你怎么知道在整个时间箱的最后找完了所有的缺陷。你不知道。当你在整个时间箱的最后你怎么能知道你覆盖了所有你需要覆盖的测试条件？你不能。也就是说，你可以应用我们之前讨论的成熟的测试估算技术——基于项目管理最佳实例的技术——来保证有正确数量的会话——以及正确数量的脚本测试——而且被使用。所以，当由它的支持者在起初建议的基于会话的测试管理没有很好地匹配时，也可根据早先的讨论来进行调整使它做得更好。

### 3.9.1 管理探索性测试的案例研究

在基于因特网的项目中，我们将基于风险的分析式测试策略和一个反应式测试策略结合。反应式测试策略包括在测试执行中使用会话的探索性测试技术。分析式策略包括基于与测试计划和项目计划并行的风险分析预先设计开发测试脚本。在测试执行中，测试经理和3个测试工程师，加起来一共有超过20年的经验，进行探索性测试。测试技术员运行脚本的测试，部分测试技术员没有测试经验而其他的只有一点点。

表3-20展示了这个结果。在测试执行中，技术员在运行测试脚本的时候每天花费大概6个小时。剩下的时间，每天的3~4个小时用来读邮件、参加会议、更新缺陷报告、进行确认测试等。工程师和经理，很投入地参与其他的工作，因此每天只能花费1~2个小时来执行探索性测试。由于他们的丰富经验，你可以看到，经验丰富的测试人员是主要的缺陷发现者。

表 3-20 管理探索性测试

职 员	7 个技术员	3 个工程师 + 1 个经理
经验	总计小于 10 年	总计超过 20 年
测试文档	精确的脚本	探索性示意图
每天测试小时	42	6
发现的缺陷	928 (78%)	261 (22%)
缺陷有效性	22	44
运行的脚本	850	0
提交的输入	~5 000~10 000	~1 000
验证的输出	~4 000~8 000	~1 000

但是，如果你检查测试覆盖率，我们会看到一个不同的情况。技术员在3个月的系统测试中运行了大概850个测试脚本。这覆盖了很大的范围，很好的文档化的范围，能够为管理层提供很好的文档化的结果。探索性测试产生了很少量的清晰文档。我们不会使用之前介绍的会话报告，部分是因为我们依赖技术员来收集和脚本相关的覆盖率证据。

现在，不仅仅脚本测试覆盖较多的范围，它还包括了更多的测试时间和测试数据。我们不会衡量这个，但是我估计手写的脚本测试会导致5 000~10 000种不同的重要输入——字符串、日期、按钮等——但是探索性测试最多是那个容量的1/5。同样的，脚本测试包含了更详细的个人结果的检查。在每个小时的基础上，探索性的测试可能有相同的

有效性,但是如果我们必须产生这些会话报告,那么我们就得放慢进度,那它的有效性就会相对减小。

所以,哪个更好呢?这不是某种实验。它根本就不是一种实验;这是一种被证实的将两种方法混合起来的办法,每种方法都有不同的优点。基于每小时发现缺陷率,探索性测试是非常有效的,我们会发现很多通过脚本不能发现的缺陷。可重用的测试脚本给我们减少了很多回归的风险,较好地减缓总体风险,较好地构建自信心。总之,这是一种成功的混合方法。

事实上,最终的方法比它首次出现的时候还要复杂。在表 3-20 中,我在脚本和探索性测试中简单地创建了一个明显的分界线。事实是,区别只是程度上的不同,而不是类别上的。测试示意图越冗长,你就越倾向于脚本测试。测试示意图越简单,就越偏向于探索性测试。

在这个项目中,就像很多项目,测试技术人员会被给予自由来对测试脚本的执行进行变化。我们告诉他们,“一个测试脚本就是到达有趣的地方的地图;当你觉得哪些地方有意思,停下来四处看看。”我们在计划测试时间中包括了一些探索系统的时间——我们可以轻易地称之为“测试会话”。另外,因为测试技术人员变得越来越有经验,我们让他们开始自己写测试脚本以及执行某些探索性测试。

某些探索性测试的热衷者坚持认为探索性测试和脚本测试是完全两码事。那是个很好的市场策略,因为你必须让你的服务有别于他人。但是,事实不是那么的黑白分明。我发现探索性测试热衷者所完成的测试示意图比我和我同事写的脚本测试更加详细。事实是,测试用例的细节级别就像我们创建的时间,是你作为测试经理在你的项目中不得不关注的一个复杂问题。<sup>22</sup>

就像我在前面讨论的关于策略的问题,所有的策略是为了达到一个目标。你应当根据你的情况选择、接受以及混合正确的策略。基于会话的测试管理可以帮助你反应式测试策略和预防性测试以更平稳的方式进行混合。

### 3.9.2 综合系统问题

让我们继续处理管理综合系统测试的问题。这种系统的主要测试管理问题是分布式系统的复杂性。通常来说,不同的测试级别会在不同地点以不同的正式水平被不同的小组执行。所以,之前讨论的所有的分布式测试问题,经常是和外包问题结合在一起的,因为不同的公司也通常被包含其中。

理想的情况是,测试管理会涉及一个贯穿不同的测试级别的主测试计划。如果整个项目组遵循一个单一的、正式的生命周期模型,这个主测试计划更容易撰写和管理。

从我自己对综合系统项目的经验,我可以说这样的项目团队如果依靠高级别测试(比如系统集成测试)作为他们仅有的质量机制就会有一个可怕的风险。如果工作产品是将 5~10 个不同的项目在数月甚至是数年的工作之后简单地合并成一个不错的运行的整体,那真是不错,但事情远不是这样简单。

---

22 我在“Managing the Testing Process”一书的第 3 章的测试用例中对测试用例的详细程度有所阐述。



所以，一个贯穿整个生命周期的包括了验证和确认的正式的质量保证过程和计划是非常有用的，这个过程和计划同时包括了静态和动态测试及所有测试级别的规范化。当然，这就意味着主测试计划和所有级别的测试计划的作者们不但要在文档上合作，而且还要在质量保证计划上合作。

因为综合系统的测试有很多内在的复杂性，我倾向于在测试执行开始前解决更多的复杂性。正式的配置管理、变更管理以及发行管理计划和过程是有帮助的。这些计划和过程必须统一定义测试团队和其他项目团队之间的接触点以及交互点。这种级别的规范化是用来保证测试对象来自可控制的已知的代码库；保证在整个项目的结尾，需求、设计和代码变更在一个管理的（减少的）方法下进行；保证这个测试是依据已知的、版本化的测试项而发生的唯一途径。

综合系统天生就是复杂的，所以它们的测试环境也很复杂，我已经做过的一个测试项目，它的策划和配置测试环境就需要差不多 10 个人一年的工作量，这让它比其他项目要大很多。它所需要的努力也是高技能和高复杂性的工作，意味着错误会导致痛苦的负面影响和项目的延迟。

不仅仅是这样，综合系统还会处理复杂的数据。为了让我们的测试结果有意义，我们需要具有相同的复杂程度的测试数据。这个就其本身而言也可以成为一个小型项目。在管理这些测试工作量的时候，我们不得不在很多张表中创建书面的上万条测试记录，这些表中有复杂的关键关系，包括名字、地址和身份号码。

某些 RBCS 的银行领域的客户可能会有很多的系统共享和管理相关数据。他们想要使用生产线数据来做测试，但是这样做会面临严重的隐私问题。对这些数据类进行匿名处理会有很大的工作量。

很多次我都说到了 IVR 项目，这是个复杂的综合系统项目。你可能会想起我们关于“主干集成”测试的讨论。在那种情况下，我们在执行系统集成测试，这些测试可能会在这些组成的系统存在前进入综合系统。所以我们要做的就是使用不同的桩、驱动、用具和模拟器。这些模拟器中的某些是很复杂的而且需要巨大的工作量来构建。

我们的一个客户是美国太空联盟（United Space Alliance），这个公司除了其他事情，也创建了很多仿真模拟器用来训练航天飞机的宇航员和执行不同的飞行前测试。这个工作需要一个完整的团队，而且这个仿真模拟器需要很深入的测试。

最后，对于大多数的综合系统来说，系统中有非常复杂的依赖关系。在很多情况下，一个系统的某些部分必需要在另外一个系统的测试运行前完成。在一个项目中，像我之前所说的一样，我不得不创建数据库用来管理这些依赖关系。<sup>23</sup>这并没有提及完全系统集成测试的需要，这些测试是横跨所有的依赖关系、接触点、共享数据目录以及其他接口。

### 3.9.3 安全关键系统问题

下面来讨论一下管理安全关键系统测试的问题。对于很多安全关键系统来说，他们

---

23 这个数据库在“Managing the Testing Process”一书的第 6 章中有描述。

的测试和质量保证需要遵循具体的工业标准。对于自动化系统来说，有 MISRA 标准，对于医疗系统来说，政府部门比如美国食品药品监督管理局使用严格的强制标准。军用系统通常来说需要不同的标准。即使你不认为是典型的安全性关键的系统，就像笔记本电脑、DVD 刻录机等，也需要遵守强制和自愿的标准。如果你有一个笔记本，翻过来看看在底部的所有标记。不能遵守这些标准意味着产品被一个或者所有的市场或者客户拒绝，不是由于法律原因就是由于某些采购检查清单上的要求被拒绝。

一个安全关键系统之所以是安全关键的是因为在失败的情况下它会直接或者间接地杀死或者伤害某人。这个风险会带来法律问题以及供应商或者创建系统的供应商的道德责任。因为不是所有的失败风险可以被降为零，我们可以使用正式的、严格的技术，包括以下这些：

- 从需求、设计和风险到代码和测试的双向追踪。
- 最小的测试覆盖级别，包括可能是多元的覆盖率，它采用我之前提到的不同的覆盖率度量方法。
- 关注于质量的验收标准，包括那些专注于非功能性的质量特性。
- 标准化、细节化以及强制的测试文档，包括结果。

这个严格性减少了很多事故疏忽、遗漏以及人为错误的风险，并且提供了符合相关规章的证据。人们可以使用这些证据来支持不同的审计，例如食品药品监督管理局在批准某个医药设备上市前进行预防性的审计；联邦航空局在飞机失事后进行反应式的审计。

和综合系统一样，由于相同的原因，我们趋向于对安全关键系统定义正式地开发生命周期。历史上，这些一般趋向于是顺序的生命周期，但是我们在医疗设备业务的一个客户使用了一个变种的敏捷开发模式（Scrum 开发模式），该设备也被认为是安全关键设备。

最后，正如早先暗示的，安全关键系统的一些非功能质量特性是需要测试的。毕竟，正确的答案给得太晚，或者正确的答案与错误的人员分享，被视为和错误答案一样危险。所以，除了性能，在测试计划中，你还应当考虑测试的可靠性、可用性，可维护性、安全性以及保密性。当然，如果安全关键系统也是一个综合系统，那么这个之前说的的问题也适用。

### 3.10 非功能性测试问题

下面继续来讨论管理非功能性测试的问题。公司有意或者无意地忽略非功能性测试是个很普遍的问题，但是那个观念非常危险。就像我之前说的一样，正确的答案给得太晚，或者正确的答案与错误的人分享，可能和一个错误的回答一样危险。为什么人们忽略非功能性测试呢？

那么，如果说忽略是无意的，那就是典型的无意识事件。这个公司就是幸运的。他们从前从来没有出现过性能、可靠性或者安全性问题，而且他们也从来没有听到过或者读到过一个公司会有类似的问题。所以，忽略是一种福气——至少，在未管理的风险成为一个不被期望发生的事件前，这个事件可能是一个非常不期望的事件，或者是一个不



被期望的会削减业务的事件。

对非功能性测试的有意识忽略经常是由于很多非功能性测试的高成本。性能和可靠性测试需要复杂的类似生产线的环境，安全性测试需要昂贵的渗透测试，可用性测试需要加入可用性测试实验室和用户界面原型。

非功能性测试也可能和我们最初的想法混淆。不同种类的非功能性测试应用于不同的系统。不是所有的系统都有安全性风险；不是所有的系统都有可用性风险；不是所有的系统都有可靠性风险。你的系统有吗？为什么会有？为什么没有？风险有多严重？非功能性风险的级别和测试复杂性和成本一致吗？对于复杂的问题有复杂的答案，不像对于功能性类似的问题一样有一个更加直接的相同的答案。

为了合理地计划、设计、实施和执行非功能性测试，我们需要考虑下面这些重要的因素：

- 利益相关者的需求。
- 需要的工具。
- 测试硬件和环境的需求。
- 组织性的考虑。
- 沟通。
- 数据安全性。

我们依次来检查一下这些因素，从利益相关者需求开始。非功能性测试的一个主要挑战来自目前非功能性需求的典型状态。非功能性需求经常描述得很差或者根本没有。

可怜测试分析师和测试经理必须发现和引导利益相关者的期望，而别指望直接获得一个较好的、有条理并已经获得批准的非功能需求说明。另外，这个工作不能等在项目的最后进行。非功能性问题的较晚发现可能暗示严重的需求和设计问题，这些问题仅仅进行简单的缺陷修复和处理是很难解决的。基于这些需求，测试分析师和测试经理必须和项目利益相关者合作来辨别和评估这些非功能性风险。就像在之前讨论的一样，这个质量风险分析需要不同的观点，在这个问题上，非功能性需求的发现也需要不同的观点。

未被测试的非功能性需求引发另外一个主要的问题。你可以通过采取以下的措施来提高非功能性需求的测试性：

- 在脑海中将测试和需求联系起来。问一下自己，“我可以想出一个测试用例清晰地显示这个需求不符合吗？”这不是唯一的现实问题。它也意味着你必须使用清晰的、简单的和一贯的措辞。试着使用在你项目中有清晰定义的词语，比如说在项目或者标准词汇表中定义的。如果你要用像应当、应该或必须这类词语，要确保理解它们在需求上下文中的意义。
- 了解你的读者。记住写作沟通必须为读者所写。在分布式项目中，考虑使用读者所精通的语言级别。
- 避免模糊地需求。在避免模糊措辞时评审是很有用的。
- 使用一个标准的需求格式。这个包括不仅仅用来捕获需求的模板或者工具，也包括细节的级别。记住，在关于可追溯性的讨论中，我说过我们需要将需求和测试

用例清晰地描述，保证我们可以根据测试结果判断哪些是可以工作的，哪些是不能工作的。

- 确认可能的和合适的需求数量。对于可靠性和性能，这既是基本的和直接的——至少如果你可以让利益相关者给你一个实际的目标！对于像可用性这样的非功能性数据，这是个挑战。<sup>24</sup>

对于非功能性特性比如可维护性，事实上错误的度量可能会误导或者很危险。例如，无论是根据一个或者多个公司编码标准，在一个现存的基于 100 000 000 行的代码库中有多少行可以允许保留？正确的答案不是零。事实上，这个甚至不是正确的问题！正确的问题是，“在较多频繁维护的代码领域我们能使用什么过程来改善我们代码的可维护性以减少与编码标准的冲突？”

### 3.10.1 工具和硬件需求

在工具领域里，我们发现一个对测试经理来说既好又坏的新闻。好的是商业工具和模拟器经常可以用来做非功能性测试，半打好的工具，包括商业和开源的，支持基于网络应用的负载、性能和可靠性测试。静态和动态测试的安全性工具都存在。对于资源利用的动态分析工具也是可行的。

但是，这些工具经常对你的效率不会有额外的帮助，因为它们用了功能性的测试自动工具。确实，很多非功能性测试的形式，像性能、负载和可靠性测试，如果不使用工具就不可能有意义。

所以，作为一名测试经理，确保你的测试估计包括了工具。另外，记住很多的非功能性测试工具——因为它们针对的是一个复杂的测试问题——是很复杂的。你不可能简单地将它们交给你的功能测试员，然后说，“这里，大家用这个运行一个性能测试”，或者“嘿，简，我知道你不是一个程序员，但是请用这个静态分析工具去辨别我们代码库中所有和编码标准矛盾的代码。”

你需要知识渊博的专家来使用大多数的非功能性测试工具。这也意味着你要么慢慢地培养公司内的专家——这样就提出了你的项目能承受怎样的学习曲线的问题，要么就是引进外部的专家，在具体的测试种类和测试工具具有专长的测试顾问。

尽管有很多可获得的商业性工具可以用于很多非功能性测试，但是有时候你不得不构建自己的工具。作为一名测试经理，你要记住构建一个复杂的工具或者模拟器是一个开发的项目。你是否可以管理一个复杂的软件或者系统开发项目，或者你是否需要引入一名项目经理或者开发经理来帮助你实现它？根据亲身参与过的一个复杂的测试工具开发项目，我可以保证这个会很有挑战性。至少你可以获得你的开发经理的支持和同情，如果你已经去问过好几次的話。

不管你是购买、使用开源，或者自己构建，确保你用来测试安全关键系统的工具是合法的。对于用来生成规范化系统的所有工具需要特定的强制规范并获得批准，包括测试工具。

---

24 我在高级测试分析的配套卷讨论了它实现的方法。



非功能性测试的硬件需求对非功能性测试经常是巨大的挑战。如果任何测试主题更频繁地和沮丧的项目经理以及测试经理相关，我不确定我曾经遭遇过。

这个问题的根本是软件和系统不是有形的。因此，不像真实世界的对象，它经常不可能从一个缩小比例的模型推断出测试结果。我们在土木、机械和航天工程的同事可以依靠对材料属性很好理解的以及物理定律来构建模型并从中进行推断。软件和系统不是物理，所以他们不是用定义的属性的材料构建，也不是用物理定律来控制它们。你可以编辑一个世界让人们飞而且子弹杀不死他们——确实，很多计算机图形学动画和视频游戏程序员都可以做到。

如果你想要衡量性能、可靠性、负载响应以及人类大脑创建的这些虚拟世界，你不得不测试所有的虚拟世界，这个世界包括硬件。所以，你需要类似产品环境的测试环境用于各种现实的非功能性测试。

我几乎可以听到经理读到这些问题时的故意逃避。我以前也做过经理，所以我理解这样的情形会给测试计划和资金带来多大的挑战。但是，你可以得到的是便宜、误导的、无效的非功能性测试结果？昂贵、准确的非功能性测试结果？或者没有非功能性测试结果？

当然，人们不会喜欢那个选择，所以他们试图减少费用并且仍旧获得准确的测试结果。有时候，你可以做这些。你可以确定狭窄的时间窗用来运行最昂贵的测试。

例如，你可以租（而不是买）所需要的虚拟的用户许可证，用你的负载测试工具来执行一个满负载模拟，试图尽可能快地完成更多的测试。这个是可行的，但是确保你有程序员和设计人员的资源后备，尽可能快地修复你所发现的缺陷。这是基于不会找到任何缺陷的假设来计划、设计或执行测试的经典的最差测试实例。

还有另外一个例子。你可以决定使用一个可用性测试实验室，在一个专门的实验室来执行可用性研究以及进行用户调查，但是只要进行一次。这个是可行的，但是你会什么时候做呢？早期，在每件事情都准备好之前？或者晚点，当你发现一个主要的设计——相关的可用性缺陷可能激发重大的以及项目中的进度延迟？再一次强调，不要犯假设可用性测试不会发现任何缺陷的计划性错误。

另一种办法是公司试图通过使用生产环境来廉价地进行非功能性测试。这种方法是由一个信念引发——不管是否合理——复制产品线可能会很昂贵，所以使用生产环境本身是仅有的实用选项。很明显的，如果这个决定一旦做出，你必须在正确的时间计划测试执行，经常是在晚上或者是周末，通过精心的计划，你可以将生产环境复原到正常工作时间时的一个完全正常情况下的状态。

经典的关于在生产系统上测试的警例是切尔诺贝利核电站泄露。<sup>25</sup>如果你必须在生产中执行测试——或许因为没有任何合理价格可以获得系统的复制，比如一个核电站或者太空飞船——那么你必须非常小心。尽可能多地识别风险，并制订相应的应急计划以保证把这些风险的影响降低到可接收的级别。

通常来说在生产中频繁进行测试是个坏主意。当你和你的同事由于环境的成本而选

---

25 见 Dietrich Doerner 的 “The Logic of Failure” 一书中有关爆炸如何发生的讨论。

选择不执行非功能性测试时，你可以选择在生产环境中测试。存在的唯一问题是测试是否是经过深思熟虑的、设计好的、计划好的以及受控制的。

对于性能、负载和可靠性测试来说，和类似生产的测试环境一样重要的是一个类似生产环境的负载级别。不要出现花很多的时间和金钱复制了生产环境，结果却使用错误的负载。确保负载生成的准确和实际可行。同时也要记住，在不同库的数据数量可以对性能、负载和可靠性结果有深远的影响。在这些方面中任何的错误——现实的测试环境、现实的测试负载，或者现实的测试数据——通常会导致无效——而且可能误导的——测试结果。

对于测试执行来说修改测试数据，或者将一个或多个主服务器共同应用在一个不可用的、奇怪的或者未知的状态是很普遍的。测试人员在面对这些问题时的通常做法是重启服务器来还原正常的操作，重新载入测试数据以及重启应用。

在测试中改变通信设施并不常见。但是，这也是可能发生的。一个路由器、交换机或者防火墙可以因为测试而受影响。

在某些情况下，修改一部分运行的通信设施用于执行一个测试也是很有必要的。例如，有些防火墙基准测试经常需要非典型的流量和网络设置。

如果需要为了一个测试而改变通信协议，那么你需要使用正确的工具。如果测试突然之间由于一个微小的、没什么了不起的改变阻塞了网络上的路由而没有成功开始，那么，在重新进行非功能性测试之前，你可能需要首先消除该负面影响。因为非功能性测试很昂贵，最后你需要做的事情是识别结果中的伪正确。最后，如果你为了执行一个测试必须要对基础设施做修改，那么确保测试人员知道如何可靠和完全地撤销该修改。

### 3.10.2 公司和安全性考虑

似乎测试环境的问题还不够复杂，我们还必须考虑组织方面的问题。如果我们要测试一个由不同的组件组成的复杂系统，而且这些组件的安放位置可能不太合理。事实上，它们可能按照地理位置和组织进行分布。例如，当我们测试一个房屋净值贷款系统时，用于处理贷款项目的一个后台系统坐落在非现场。另一个后台系统属于另一个公司，一个信用社，也坐落在非现场。我们不得不小心地计划和协调我们和信用社的接口测试。如果我们遵守规则，测试就会进行得平稳一些。但是，对于贷款处理系统来说，我们很难去说服这些人——它们是同一个公司的职员——在测试过程中不要做像打补丁、重启系统、开启带宽和其他消耗后备导致测试混乱的事。

在这些环境中，如果有测试环境或者测试支持人员的可用性问题的话，那么在计划中要确保考虑到这些。忘记计划哪怕只是一个基本的后台或者下线系统——不管它多简单——都可能严重地影响进度。

非功能性测试的最后一个问题就是——这个问题适用于所有种类的测试——即确保将安全性措施列入你要执行的测试，并制订相应的计划。例如，在美国，支付卡工业数据安全（PCI DSS）标准需要不同级别的数据加密。这个包括交易加密——例如，对于传输信用卡信息的无线路由使用 WPA 加密——就像数据在目录中一样——例如，将存储敏感信息的数据库区域加密。显然地，如果是测试安全性，我们需要检查是否和这个



标准相符。但是，如果我们试图测试功能以及将期望结果和数据值进行比较，加密会让它变得困难。

如我之前所说的一样，对于性能、负载和可靠性测试来说使用实际数据集是非常必要的。将同类错误的数据包含在真实数据集里也是非常有用的。除了使用生产数据，没有更好的办法来执行这一过程。但是，做这些可能会使我们触犯各种法律、标准或者公司的安全规范，这些规范用来阻止敏感数据的泄密。在决定了外包或者内包测试后，这个问题可能会由于应用于非本公司雇员的不同规则而突然爆发，特别是如果他们还在不同的国家。

在某些情况下，公司试着用匿名测试数据来对付这些问题。换句话说，他们创建一些工具或者算法允许他们使用可逆的、安全的方法获取生产环境的数据。这样做面临两个巨大的挑战。第一，获取的生产数据必须保证测试数据看起来仍然像生产数据。例如，姓名应当仍旧看起来像名字，而不是一串随机的字符，地址潜在地应该是有效（尽管是虚假的）地址、兄弟姐妹、活着的双亲，以及孩子的数量经常是真实的等。要不然，整个目标就不能实现。其次，获取数据时必须捕获在各个表中共享的重要点，这样在不同表中记录之间微妙的关键联系才得以保存。如果不是，那么从数据集之间的相互关系来看，你可能不会在最重要的方法中使用看起来像生产数据的测试数据。

在测试领域，记住，事先计划总归比意外好。如果在测试中你发现安全性相关的问题，请将这个问题提交到合适的级别进行解决——或者接受测试范围的减少。我有一个在银行领域的客户接受的事实就是，他们不能查找很多与数据复杂性和数据质量相关的缺陷，因为他们的公司安全官员不让他们使用生产性数据进行测试。

### 3.11 认证考试模拟题

在每个章节的最后，我们会准备一个或多个考试习题来加强你对该章内容的理解并用来准备将来的 ISTQB 高级测试经理考试。

1. 作为一名测试经理，下列哪些步骤可以减轻项目风险？
  - A 测试性能相关的问题
  - B 在主要测试分析师离去后雇佣一个合同工
  - C 建立一个额外测试环境应对测试常用环境的失败
  - D 使用测试结果进行项目回顾
2. 您在策划由 3 个现货组件集合而成的系统的测试，该系统用来管理银行的一个应收款系统。您在组织项目和系统利益相关者之间的一个非正式质量风险分析会议以决定哪些测试条件应该被测试以及每个测试条件应当被测到什么程度？指出在这个质量风险分析会议中识别出的质量风险项。
  - A 组件供应商不能执行充分的组件测试
  - B 计算账单支付滞纳金
  - C 对国际供应商的支票及时支付
  - D 使用可能性和影响程度计算风险优先级

3. 在遵循失效模式和影响分析技术的正式风险分析会议中, 当您在计算风险优先级时, 下面哪些是计算中的主要因素?
- A 严重性和优先级
  - B 功能性、可靠性、可用性、效率、可维护性以及可移植性
  - C 测试团队主要贡献者数量的减少
  - D 开发团队主要贡献者数量的减少
4. 假设您是一名负责银行的集成测试、系统测试和验收测试的测试经理。现在的项目用来将一个现有的 ATM 系统升级到该系统可以允许顾客从支持的信用卡获得现金预付款项。这个系统应当允许从\$20~\$500 的现金预付款项, 包括支持所有信用卡。所支持的信用卡有 American Express、Visa、JCB 卡、Eurocard 和 MasterCard。
- 在一个质量风险分析中, 下面哪一个陈述可以最好地将利益相关者和他们的输入联系起来?
- A 一个测试人员可以提供对风险项可能性的输入
  - B 一个开发人员可以提供对风险项影响程度的输入
  - C 一个业务分析员可以提供关于风险项可能性的输入
  - D 一个服务台职员可以提供关于风险项影响程度的输入
5. 下面哪种情况下, 您期望一个迭代性的质量风险分析以获得最大数量的新建或者改变风险项和风险级别?
- A 在执行一个最终需求规格说明的风险分析之后得到设计规格说明的一个草稿
  - B 一名测试人员在测试设计完成后离开, 然后您雇佣了一个新的测试人员取代她
  - C 开发经理在质量风险分析结束的时候另外雇佣两个程序员
  - D 对最终的需求规格说明进行风险分析后, 文档被放在正式配置管理系统中
6. 假设您是某项目中的一名测试经理, 该项目用来创建一个可编程恒温器用来控制家用的中央供热系统, 空气流通设备和空调系统。除了正常的 HVAC 控制功能, 这个温度控制器也能够下载数据到运行于 PC 中的一个基于浏览器的应用以便进行后续分析。
- 在质量风险分析中, 您识别出了基于浏览器应用和 PC 配置间的兼容性问题, 该问题是作为一项质量风险, 发生的可能性很大。针对该风险您计划执行兼容性测试。
- 下面哪种方法可以用来监视降低该风险的测试效果?
- A 减少支持的 PC 配置数量
  - B 分配更多的测试人员来覆盖兼容性而不是覆盖功能性
  - C 分析和该风险项相关的已经发现的缺陷数量
  - D 计划测试更多通用的 PC 配置
7. 下面哪些项目实例使用失效模式和影响分析方法进行风险分析会更好?
- A 项目团队第一次应用基于风险的测试
  - B 测试系统不仅复杂, 而且是安全关键的
  - C 测试系统是一个财务系统
  - D 把减少文档数量当做主要问题的项目
8. 假设您是一名负责银行的集成测试、系统测试和验收测试的测试经理。现在的项目用



来将一个现有的 ATM 系统升级到该系统可以允许顾客从支持的信用卡获得现金预付款项。对所有支持的信用卡而言，这个系统应当允许客户提取从\$20~\$500 的现金预付款项。所支持的信用卡有 American Express、Visa、JCB 卡、Eurocard 和 MasterCard。在主测试计划中，待测试部分的特性如下：

- I. 所有支持的信用卡
- II. 语言本地化
- III. 有效及无效的预付款
- IV. 可用性
- V. 响应时间

仅根据以上所给的信息，选择已有足够充分的信息可以开始测试设计的待测试特性。

- A I
- B II
- C III
- D IV
- E V

9. 续上题中描述的场景。下面哪个话题需要在主测试计划中详细描述？

- A 一个回归测试策略
- B 一个预付款数额边界值列表
- C 一个测试用例相互依赖关系的描述
- D 测试用例的一个逻辑组合

10. 假设您是某项目中的一名测试经理，该项目用来创建一个可编程恒温器用来控制家用的中央供热系统、空气流通设备和空调系统。除了正常的 HVAC 控制功能，这个温度控制器也能够下载数据到运行于 PC 中的一个基于浏览器的应用以便进行后续分析。

这个公司的测试策略是每个测试用例要在配置选项的所有组合上运行。对于这个系统来说，识别出下面的各因素以及每个因素下的各选项：

- 支持的 PC/恒温器连接：USB 和蓝牙。
- 支持的操作系统：Windows 2000, Windows XP, Windows Vista, Mac X, Linux。
- 支持的浏览器：Internet Explorer, Firefox, Opera。

因为有 10 个测试用例和数据下载相关，这就需要运行 300 个测试用例，每个用例需要运行 1 个小时。

有了管理层的批准，您决定测试 5 个配置，覆盖了每个选项但没有覆盖所有可能的两组和 3 组选项组合。

下面哪一项最适合在撰写和测试策略偏离的文档时使用？

- A 在测试设计规格说明中，解释项目计划的不同方法以及如何建立测试配置
- B 在测试过程规格说明中，解释应当和某个配置对应运行的测试用例
- C 在主测试计划中，解释这个项目计划的不同方法以及为什么这些方法是充分的
- D 在测试项传递报告中，解释项目计划的不同方法以及和某个配置对应运行的测试

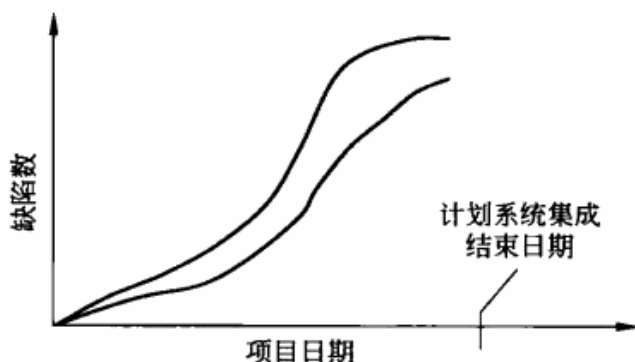
## 用例

11. 续上题中描述的场景。您在写一个主测试计划用来覆盖可编程恒温器的集成测试和系统测试。选择以下所有正确的表述。
- A 方法部分应当描述如何测试恒温器和 HVAC 系统其他部分的集成
  - B 进度部分应当表述集成测试应当何时开始以及系统测试应当何时开始
  - C 环境需求部分应当描述对测试的每个级别负责任的人员
  - D 测试项部分应当描述测试每个级别所需要的设备
  - E 测试交付部分应当描述测试每个级别所汇报的结果
12. 续上题中描述的场景，您是一名项目的测试经理，该项目用来创建一个可编程恒温器用来控制家用的中央供热系统、空气流通设备和空调系统。该系统中的一个可识别的关键质量风险是由于压缩机过度运转导致 HVAC 系统可能的破坏（例如，在短时间内重复地开启和关闭该单元）。下面哪一些方法是用 IEEE 829 测试计划来指导这个风险的合适测试方法？
- A 对这个测试级别单独编写测试计划
  - B 列出预防过度运转的特性并对此特性进行测试
  - C 在测试计划的介绍中详细列出可编程恒温器的所有需求
  - D 加入一个全功能压缩机作为一个测试项
13. 续上题中描述的场景。
- 从历史上看，在过去的 7 个项目中，测试团队每人每月发现了大约 12 个缺陷。5 名开发人员被派去从事一个计划持续 6 个月的新项目。假设累计发现的缺陷数量，正如您的覆盖率表中所发现的，已经达到 351 个缺陷。
- 基于这些仅有的信息，选出以下最有可能正确的一些项？
- A 你可能期望在系统测试结束前找到超过 20 个缺陷
  - B 你至少忽略了一个关键质量风险目录的测试
  - C 你需要一个至少由 3 个测试人员组成的测试团队进行最优测试
  - D 你已经获得系统测试中发现缺陷的大概数量
14. 续上题中描述的场景。
- 假设这个项目遵循迭代的生命周期模式，而您之前有缺陷度量的项目遵循顺序生命周期模型。假设该项目和前期项目没有其他不同，那么以下哪些可能是影响预测缺陷数量准确性的原因？
- A 人员因素
  - B 材料因素
  - C 过程因素
  - D 质量因素
15. 您是一名负责一个项目系统测试的测试经理，这个项目是在一个新型小汽车中更新定速巡航模块。这个定速巡航软件更新的目标是让小汽车更具有燃油经济性。
- 根据最终的需求规格说明，您已经写出系统测试计划的第一个版本。您收到了设计规格说明的一个早期草稿。以下选项正确的是



- A 在最终版本的设计规格说明成型前不要更新系统测试计划
  - B 基于设计规格说明的这个版本产生系统测试计划的一个草稿更新
  - C 检查设计规格说明的这个版本和需求规格说明的不合规性
  - D 参加设计规格说明的最终评审，而不参加任何设计规格说明早期草稿的评审
  - E 检查质量风险分析确认设计规格说明是否标识了额外风险项
16. 从质量风险的剩余级别来看，下面哪些技术在控制测试过程时最好？
- A 计算发现的缺陷数量和解决的缺陷数量
  - B 计划成功的测试用例数量和失败的测试用例数量
  - C 计算执行通过的需求数量和已知缺陷的需求数量
  - D 计算未知缺陷测试风险项的数量和已知缺陷测试风险项的数量
17. 您是一名负责一个项目系统测试的测试经理，这个项目是在一个新型小汽车中更新定速巡航模块。这个定速巡航软件更新的目标是让小汽车更加具有燃油经济性。在测试执行的中途，您发现根据测试结果无法判断是否已经改进了燃油经济性。识别以下所有您可能指导测试分析人员来执行的措施并帮助解决这个问题。
- A 重新设计燃油经济性测试
  - B 修改质量风险分析
  - C 修改测试环境以收集更加详细的实际结果
  - D 检查已测试的燃油混合物的稳定性
  - E 没有明显的改变时仍旧报告该车具有较高燃油经济性
18. 假设您是一名负责银行的集成测试、系统测试和验收测试的测试经理。现在的项目用来将一个现有的 ATM 系统升级到该系统可以允许顾客从支持的信用卡获得现金预付款项。对所有支持的信用卡而言，这个系统应当允许客户提取从\$20~\$500 的现金预付款项。所支持的信用卡有 American Express、Visa、JCB 卡、Eurocard 和 Mastercard。在测试执行中，你发现 5 个缺陷，分别是由不同测试人员所报告，都是和现金预付款一样的问题，在这些报告中唯一的不同是被测试的信用卡。你可能建议采取下面哪些措施来改进测试过程？
- A 修改所有现金预付款测试用例使其只用来测试一个信用卡
  - B 检查所有后续存档的报告，在将其交付给开发人员前关闭所有重复的缺陷报告
  - C 改变需求，删除对 American Express 的支持
  - D 让测试人员检查其他卡的相似问题并在缺陷报告中汇报
19. 您是一名银行的质量评估小组的经理，负责银行项目的独立测试。您的项目要使用 3 个现成系统集成的综合系统来管理银行应收款系统。您现在负责执行系统集成测试。考虑以下缺陷打开/关闭或者覆盖率图。
- 下面对这幅图的解释中，哪些项为声明系统集成测试未完成提供了理由？
- A 缺陷发现率还不稳定
  - B 开发人员解决缺陷不够快
  - C 测试集还没有被完全运行

D 很多缺陷没有解决



20. 下面哪些项是内部失效成本的例子?
- A 在测试中发现一个缺陷
  - B 培训开发人员进行安全编码练习
  - C 设计测试用例
  - D 解决客户发现的缺陷
21. 您是一名银行的质量评估小组的经理, 负责银行项目的独立测试。您的项目要使用 3 个现成系统集成的系统来管理银行应收款系统。您现在负责执行系统集成测试。下面哪些项最可能成为测试这一系统的主要商业动机?
- A 避免生命损失
  - B 改正客户订单时更有信心
  - C 在发行前找到尽可能多的缺陷
  - D 搜集证据控告组件承包商
22. 下面哪些项是外包测试的风险而不是分布式测试的风险?
- A 选择一个不合适的测试承包商
  - B 由时区差别引起的沟通问题
  - C 测试团队成员的不熟练技能
  - D 在不同测试位置测试过程的不一致
23. 您是一名银行的质量评估小组的经理, 负责银行项目的独立测试。您使用质量风险分析在测试准备中分配工作量以及对您的测试设定优先级。现在要执行系统集成测试。
- 您已经对每个测试用例都运行了一次, 但不肯定项目管理团队会给您多少时间进行额外的测试执行, 因为监管的变化可能需要在时间表之前将系统激活。
- 基于每个测试用例所发现缺陷的严重性和优先级, 可以计算每个测试用例失效的权重。根据测试用例覆盖的质量风险项, 可以知道每个测试用例风险优先级数字。考虑测试失效的加权和风险优先级数字, 可以在最后几天或者几个星期的测试中重新对测试用例进行优先级划分。
- 下面哪些项是优先级重新划分的优点?
- A 如果测试被缩短, 那么花费的工作量可能会最小
  - B 如果测试被延长, 那么可能会有时间运行所有的测试



- C 如果测试被缩短，那么可能会运行最重要的测试
  - D 如果测试被延长，那么可能会覆盖所有的需求
24. 下面哪些项是探索性测试和其他反应式测试策略的优点，而基于需求分析的测试策略却没有？
- A 使用一个非常有经验的测试团队的能力
  - B 在发布前准确预测剩余风险的能力
  - C 在需求分析中预防缺陷的能力
  - D 在没有完整的测试基础的情况下进行有效测试的能力





**D-D 路径：**未在 ISTQB 词汇表中定义。

**数据流分析：**一种基于变量定义和使用的静态分析形式。

**决策表测试：**一种黑盒测试设计技术，测试的设计用例用来测试判定表中各种条件的组合。

**决策测试：**白盒测试设计技术的一种，设计测试用例来执行判定结果。

**基于缺陷的技术：**参见基于缺陷的测试设计技术。

**基于缺陷的测试设计技术：**针对已知的某个种类的缺陷，设计测试用例来测试该类别的缺陷的过程。

**缺陷分类法：**一种缺陷分类（等级）系统，用于缺陷分类。

**动态分析：**组件或系统的执行过程中对其行为评估的过程，例如对内存性能、CPU 使用率等的评估。

**错误推测：**根据测试人员以往的经验，猜测在组件或系统中可能出现的缺陷以及错误，并以此为依据来进行特殊的用例设计以暴露这些缺陷。

**等价类划分技术：**黑盒测试设计技术，该技术从组件的等价类中选取典型的点进行测试。原则上每个等价类中至少要选举一个典型的点来设计测试用例。

**探索性测试：**非正式的测试设计技术，测试人员能动地设计一些测试用例，通过执行这些测试用例和测试中得到的信息来设计新的、更好的测试用例。

**基于经验的技术：**参见基于经验的测试设计技术。[注意：该词汇表错误地将此叫做一个“基于经验的技术。”]

**基于经验的测试设计技术：**根据测试人员的经验、知识和直觉来进行用例设计和/或选择的一种技术。

**LCSAJ：**线性代码序列和跳转。包含以下 3 项（通常通过源代码清单的行号来识别）：可执行语句的线性序列的开始、结束以及在线性序列结尾控制流所转移到的目标的行。

**内存泄漏：**程序的动态存储分配逻辑存在的缺陷，导致内存使用完毕后不能回收而不可用，最终导致程序因为内存缺乏而运行失败。

**复合条件测试：**一种白盒测试设计技术，测试用例用来覆盖一条语句中的单条件所有可能的结果组合。

**成对测试：**一种黑盒测试设计技术，被设计出来的测试用例能执行每个成对输入参数所有可能的离散组合。

**路径测试：**一种白盒测试设计技术，设计的测试用例用以执行路径。

**基于需求的测试：**根据需求推导测试目标和测试条件以设计测试用例的方法。例如，执行特定功能的测试或探测，诸如可靠性和可用性等非功能属性的测试。

**软件攻击：**参见攻击。

**攻击：**直接和集中地通过尝试强制产生特定失败来评估测试对象的质量，特别是其可靠性。

**基于规格说明的技术：**参见黑盒测试。

**黑盒测试：**不考虑组件或系统内部结构的功能或非功能测试。

**静态分析：**分析软件工具（如需求或代码），而不执行这些工作产品。

**语句测试：**一种白盒测试设计技术，所设计的测试用例用来执行语句。

**状态转换测试：**一种黑盒测试设计技术，所设计的测试用例用来执行有效和无效的状态转换。

**基于结构的技术：**参见白盒测试设计技术。

**白盒测试设计技术：**通过分析组件/系统的内部结构来产生和/或选择测试用例的过程。

**测试章程：**对测试目标的陈述，还可能包括关于如何测试的测试思路。测试章程通常用在探索测试中。

**用例测试：**一种黑盒测试设计技术，所设计的测试用例用于执行用户场景。

**野指针：**一个指针，所指的地方在这个指针的范围之处或不存在。



## 软件特性测试

守桥人：停下！任何人过生死桥前必须回答我的3个问题。

鲁宾爵士：问吧，守桥人。我不怕。

守桥人：你叫什么？

鲁宾爵士：卡麦罗特的鲁宾爵士。

守桥人：你找什么？

鲁宾爵士：我找圣杯。

守桥人：亚述首都叫什么？

鲁宾爵士：我不知道。【尖叫着从桥上掉下去】

守桥人：你叫什么？

格雷德爵士：卡麦罗特的格雷德爵士。

守桥人：你找什么？

格雷德爵士：我找圣杯。

守桥人：你最喜欢的颜色是？

格雷德爵士：蓝色。不，是黄……【尖叫着从桥上掉了下去】

守桥人：【大笑】停下！你叫什么？

亚瑟王：不列颠国王的亚瑟。

守桥人：你找什么？

亚瑟王：我找圣杯。

守桥人：轻燕的飞行速度是？

亚瑟王：你问的是非洲的燕子还是欧洲的燕子？

守桥人：我不知道。【尖叫着从桥上掉了下去】

乡绅：你怎么知道那么多关于燕子的事？

亚瑟王：知道吗？当你是亚瑟的国王时你必须知道这些。

——在一部由英国巨蟒喜剧团写的怪诞古典影片《巨蟒与圣杯》中  
亚瑟王展示了他对飞行特性的精通。

高级大纲的第5章是讨论基于在ISO 9126中描述的质量特性分类所进行的软件特性的测试。本章的概念主要适用于测试分析师和测试技术分析师。对于测试经理而言，本

章没有任何级别的学习目标。但是，因为测试经理管理测试分析人员和测试技术分析人员团队，最好要熟悉他们工作的概念和术语。我建议测试经理阅读一下高级大纲的第5章。

另外，如果你在为 ISTQB 高级测试经理考试做准备，记住和软件特性测试相关的一些概念，特别是 ISO 9126 标准，这在基础级大纲中也讲到而且是在考试范围之内。所以，仅仅出于熟悉和回顾这些内容，你也要阅读高级大纲的第5章。同时也应当回顾基础级大纲第2章的第三小节，保证你已经掌握那章的所有学习目标。

### ISTQB 术语

**可达性测试：**可达性测试是指测试残疾人或不方便的人们使用软件或组件的容易程度。

**准确性测试：**在 ISTQB 术语表中没有定义。

**准确性：**软件产品提供的结果的正确性、合规性和精确程度的能力。

**效率测试：**确定测试软件产品效率的测试过程。

**启发式评估：**一种静态可用性测试技术，判断用户接口和公认的可用性原则的符合度。

**互操作性测试：**判定软件产品可交互性的软件测试过程。

**可维护性测试：**判定软件产品可维护性的测试过程。

**运行验收测试 (OAT)：**验收测试阶段的运行测试，由操作员或管理员在一个模拟实际运行环境的系统中执行，关注软件运行方面的行为。例如：可恢复性、资源行为、易安装性及是否符合技术标准。

**运行概况：**由组件或系统执行一系列不同任务的代表值，可能基于与组件系统交互时的用户行为以及它们发生的概率。任务是逻辑的而不是物理的，所以能跨多个机器执行或在非相邻的多个片段执行。

**可移植性测试：**判定软件产品可移植性的测试过程。

**可恢复性测试：**判定软件产品可恢复性的测试过程。

**可靠性增长模型：**通过对软件或系统不断测试，并去除其中的缺陷，可靠性不断增长的模型。

**可靠性测试：**判定软件产品可靠性的测试过程。

**安全性测试：**判定软件产品安全性的测试，参见功能性测试。

**适用性测试：**在 ISTQB 术语表中没有定义。

**适用性：**软件产品为特定功能和用户目标提供一套合适功能的能力。

**软件可用性度量调查表 (SUMI)：**一种基于调查表的可用性测试技术，以评估组件/系统的可用性，如用户满意度。

**可用性测试：**用来判定软件产品的可被理解、易学、易操作和在特定条件下吸引用户程度的测试。



# 评 审

根据周四发布的评审报告，美国宇航中心损失了价值 1.25 亿美元的火星探测器，原因是在做一个关键飞行数据的操作时，Lockheed Martin 的工程师们用了英制而另外一个团队用了公制。度量单元的不同阻碍了 Lockheed Martin 火星气候探测器空间团队与位于加州 Pasadena 的美国宇航中心飞行试验室的飞行团队之间跟踪信息的交换。

——CNN 报道。一个新的关于缺陷的故事，这个缺陷本可以通过设计评审来发现。

高级大纲的第 6 章主要讲述评审。就像在基础大纲中讲解的一样，评审是静态测试的一种，它依靠人员而不是工具来分析项目或者一种项目工作产品，比如说需求规格说明。通常来讲，评审的主要目标就是找出缺陷并在该工作产品作为后续项目行为的基础进行拓展前消除缺陷，当然它还能实现其他的目标。高级大纲还介绍了几种额外的评审和涉及可以进行更有效和成功的评审的策略。高级大纲第 6 章有以下 5 个部分：

1. 概述。
2. 评审的原则。
3. 评审的类型。
4. 引入评审。
5. 评审的成功要素。

下面依次来看每个部分以及它们和测试管理之间的联系。

## 6.1 概 述

### 学习目标：

能回想起该部分内容即可。

为了成功地进行评审，一个组织必须对计划、参与和跟踪进行投资，并保证这些活动的质量。一个成功的评审可不仅仅是一屋子的人读一篇文档。

好的测试人员也是优秀的评审员。好的测试人员时刻保持好奇心并且善于提出问题，就像在基础级大纲中我们讲的“有经验的悲观情绪”。与只是想着如何实施项目的开发人员和其他项目人员相比，好的测试人员会提出不同的技术观点。测试人员倾向于思考软

件可能怎样被破坏。这样的观点使测试人员更适合做评审，但是需要他们的出发点更积极。

这不仅仅是对测试人员的要求，也是对所有参与人的要求。因为评审是一个团队工作，所有参与者必须全身心地投入其中，以保证评审顺利执行。一个消极的、武断的参与者会对整个工作进程造成严重的影响。

评审工作是很难做好的，所以对很多组织而言，如果一开始评审工作没有做好的话，他们常常就会放弃。但是，如果被合理地执行，评审将是所有与质量有关的活动中性价比最高的工作。

## 6.2 评审的原则

### 学习目标:

(K2) 解释评审相比动态测试和其他静态测试技术的有利面。

#### ISTQB 术语

**评审：**对产品或产品状态进行的评估，以确定与计划的结果所存在的误差，并提供改进建议。例如，管理评审、非正式评审、技术评审、审查和走查。

**评审人：**参与评审的人员，辨别并描述被评审产品或项目中的异常。在评审过程中，可以选择评审人员从不同的角度评审或担当不同的角色。

我们来探讨一些评审原则。这些原则在基础大纲中也已经解释过了。

首先，评审是静态测试的一种。所以，被评审的目标在评审期间不会被执行或运行。像任何测试行为一样，评审可以有不同的目标。通常的目标是寻找缺陷。其他的目标，是测试所特有的，评审可以为我们项目的继续进行建立信心，减少相关的风险，生成对项目团队和管理层有用的信息。某些独特的评审还有两个额外的共同目标。一个是保证对文档和项目的内在含义有同样的理解，另一个是在文档说明上达成一致。

评审经常会在动态测试前进行。评审应当作为动态测试的补充。缺陷在系统中停留的时间越长，缺陷成本增加越多，所以评审应该尽快进行。但是，不是所有的缺陷都可以在评审中很容易地被发现，所以动态测试还是需要的。

除了评审以外，静态测试的另一种形式是静态分析。就像所有的静态测试一样，静态分析不会执行被测试件。但是，静态测试使用工具来检测缺陷。评审运用评审人员的智力来发现缺陷。静态分析工具都很有条理，稳定，不会像人脑那样感到无聊和枯燥，同时他们也都严格执行，这些都是评审者所不具有的特点。所以单调的任务，例如评审数百万行代码是否和编码准则有冲突，或者评审上百页的规范说明文档是否有拼写、语法以及阅读困难问题等，可以而且应当通过静态分析来解决。

据报道，Woody Allen，一个纽约的电影导演，曾说过“80%的成功来源于自我表现”。可能在电影工业里这是正确的，但是 Woody Allen 不会是个成功的评审人员。评审需要



充分的准备。如果你没有花费时间在准备评审上，你也不会评审会议上获得价值。

事实上，如果你在会议上问很多愚蠢的问题，就很容易把会议变得毫无价值，其实提问前你只要仔细阅读一下文档自己就能回答那些问题。你或许认为那不是个很棘手的局面，特别是鉴于著名的管理方案，即“根本就没有愚蠢的问题”。不过，确实有很多的蠢问题。在会议上如果有某人的问题是因为他们自己没有好好准备，从而导致整个屋子的人不得不看着其中一个人花时间来指导这个未准备好的与会者，而这些问题他在进入会议室前就应当知道。这些问题都是愚蠢的问题。事实上，在评审会议上没有准备的出现对我来说就是一种很粗鲁的行为，不尊重会议室里其他人的时间。

只要执行得当，评审是非常有效的，公司应当评审所有重要的文档。文档包括测试文档：测试计划、测试用例、质量风险分析、缺陷报告和测试状态报告等。我推崇的原则是，任何重要的事情在需要至少被两个人看完之后才可以算结束。你不必要评审不重要的文档，但是你的问题是：为什么要写一个不重要的文档？

那么，评审结束之后会发生什么？可能会有 3 个结果。理想的情况是该文档很好或者只是有一些小的变动。另一种可能是文档需要做些改动但是不需要重新评审。最费事的结果就是（从工作量和时间表来说）该文档需要大量的改动以及重新评审。现在，如果真的发生了这种事情，记住，虽然这是一个很费时的结果，但是这比起如果你忽略这些严重的问题以至于只能在组件测试、集成测试、系统测试，甚至验收测试的时候进行处理，相对来说还是可以节省很多金钱和时间的。

## 6.2.1 正式和非正式评审

在一个非正式评审中，没有定义明确的规则、没有定义明确的角色、没有定义明确的工作职责，所以你可以随意按照自己的意愿来执行该评审。当然，请记住 Capers Jones 指出非正式评审通常只能找出大约 20% 的缺陷，而正式的评审可以找出高达 85% 的缺陷，比如审查。如果某项任务很重要，你可能会想要执行一次正式评审——除非你认为你和你的团队聪明到不会犯任何错误。<sup>1</sup>

一个正式评审应当包括以下基本的角色和职责：

- 经理：经理主要负责资源分配、安排评审等。但是，他们可能不允许参加某些特定的评审会议。
- 主持人或者主管：此人是负责评审会议的主席。
- 作者：此人是待评审内容的作者。一个合适的评审会议不应该成为作者一段糟糕或者耻辱的经历。
- 评审人员：此人负责评审工作，也就是在评审中找出缺陷。按照他们的专业技能或者根据其关注的缺陷种类，评审人员可以担当特别的职能。
- 会议记录员/秘书：此人负责记录评审工作中的发现。

现在，在某些评审中，可以身兼数职。比如，有人可以同时担任作者、主管以及秘

---

<sup>1</sup> 参阅 Capers Jones 的“Software Assessments, Benchmarks, and Best Practices”一书。我们会在本章节的后续部分进行仔细讨论。

书。事实上,作为一个测试经理,当让我自己的测试团队来评审我的测试计划时,我既是经理,同时也是作者、主持人以及秘书。

一次评审也可能会需要其他职能人员的参与。我们可能需要加入决策者或者项目利益相关者。如果评审的次要目标甚至主要目标是为了进行信息发布以及保证各方观点一致,那就更需要更多相关人员的加入。在某些情况下,我们还可以要求客户或者用户代表参与进来。最近我们就对我们新网站的实物模型做了一次评审,设计人员包括我们的市场团队、网站开发外包合作团队,以及公司执行层。由于我们雇佣了该网站开发团队,对他们而言我们是客户,而对该网站的特性来说,我们是用户。

类似的较正规的评审通常还需要一位阅读人员,此人负责阅读测试项并进行解释说明。

你也可以在一个公司中使用多种评审类型。对于某些文档来说,是否准时比是否完美更加重要。例如,在我们的测试团队中,我们运用“两双眼”规则保证测试人员的缺陷报告在提交前必须由另一位测试人员看过。但是,对于更加形象的文档,像是测试计划,我们会在整个测试团队中用走查方法。对于关键文档而言,你可以对单个测试项运用一种以上的评审类型。

## 6.2.2 非正式评审的实例研究

图 6-1 描述了软件开发副总裁对我们的测试计划草案进行一次非正式评审之后的反馈。我选择了他邮件中 10 个左右的评论来归纳其反馈。让我们来看一下,在图 6-1 中我们获得了哪些好处。

我检查了最新版本的计划,提供了以下建议:……

2. 在服务器测试领域中,我们应当修改测试计划对低于 500 000 个用户的情况进行测试。在早期的生产环境中我们不会有支持 500 000 个用户的硬件,可能会先支持 100 000 个用户。在开始阶段我们也只会拥有 25 000 个邮箱……

4. 出口准则。[硬件开发副总和我]应当制定一些规则。我相信,当一个系统有 25 000 个顾客执行邮件、浏览等,并且在某段时间内 24 小时不间断运行,我们需要查看一些运行准则。我觉得我们至少需要一周时间……[另外],我需要弄清楚运行时间的需求,比如执行一轮所有系统测试套件需要多少时间。系统测试阶段需要执行几轮测试。

5. “必须修复”的[缺陷的定义]——我承认[这是必需的],而且我写下了一些笔记来[解决这个问题]……

8. 发布版本说明。这点提得较好。从[测试发布管理]来看,我们没有把工作做好。我们需要开始为每个版本提供版本以及描述哪些代码废弃了,哪些新功能可用了,以及哪些缺陷修复没加入。

图 6-1 实例:评审的优点

第二项,发布时的硬件局限——首次发布时的极限用户是 100 000 这一条,为我们节约了开发支持 500 000 个用户的负载生成器和测试脚本的大量工作。这些信息不在市场的需求文档里,但是副总裁知道这一点。这个例子说明,需求规格说明中的一个缺陷导致测试计划的缺陷,幸运的是它在本次评审中被捕获到了。

第四项为大家对系统可靠性的需求达成共识很有好处。这些非功能性需求在规范说



明书中经常没有很好地定义，这个项目也没有例外。所以，在这里我们使用测试计划和测试计划评审来对特定测试类型的期望结果达成一致意见。

第四项在最后也包括一个需求，即软件开发副总要求澄清运行测试所需时间。这样做的好处是项目团队上下成员对项目的重要细节形成共同的期望。

第五项指出了开发过程的漏洞，即对缺陷的分类和设定修复的优先级。以我的经验，执行上述行为最好的办法是通过使用一个跨职能部门的团队。这个团队通常叫做必须修复委员会，一个缺陷分流委员会，或者一个变更控制委员会。不管它应该叫什么，在你组织中怎么称呼，或者还没有这个团队。我在测试计划的草案中提到了这个问题，而且软件开发副总也已经开始负责处理这个问题。由于在测试执行中管理一张必须修复缺陷表格是一个关键测试过程，软件开发副总负责来解决这个漏洞被认为对测试团队和项目团队非常有益。

第八项，就像第五项，对测试也非常有益。软件开发的副总负责清除在非正式测试时我们遇到的测试发布管理中存在的某些问题。我们要求更结构化的、正规化的测试发布，以及详细的测试发布说明，这3条他都完全同意。

#### ISTQB 术语

非正式评审：一种不基于正式（文档化）过程的。

评审审查负责人（也称主持人）：负责检视或其他评审过程的负责人或主要人员。

走查：由文档作者逐步陈述文档内容，以收集信息并对文档内容达成共识。

## 6.3 评审的类型

### 学习目标：

能回想起该部分内容即可。

本节的概念主要适用于测试分析师和测试技术分析师。本章对测试经理并没有定义任何级别的学习目标。但是，如果你在学习并准备 ISTQB 高级测试经理考试和管理评审和审计有关的某些概念，以及其他在基础大纲级别的评审都是可能出现在考试中的。在准备考试的学习科目中，你要确保回顾一下基础大纲的第3章以及评审，然后读一下高级大纲中的类似部分。

不管测试经理是否参加高级测试经理考试，对他们来说，理解主要的评审类型是非常重要的。基于 IEEE 1028 标准，基础大纲讨论了4种主要的评审类型，我们会在本章后面进行讨论：

- 在正式性最低的级别（通常，对于缺陷消除有效性来说），我们会说是非正式评审。这个评审可以简单到只有两个人，作者和他的同事，通过电话来一起讨论设计上的问题。
- 技术评审更加规范化，但是还不是最正式的。有些时候它们叫做同行评审，也就

说明了典型的参与者。

- 走查通常就是指作者同时也是评审主持人对其产品日程细节进行评审。也就是说，这个作者领导这次评审，在评审中，评审人员会在每个部分浏览待评审项。
- 审查是最正式的评审。其规则被很好地进行了定义。经理可以不用参与其中。作者可能既不是主持人也不是会议记录员，而且通常需要一位阅读者。

如你所想，当正式级别上升时，评审率——每小时页面数量——会下降。

你应当记住 IEEE 1028 标准和基础大纲是讨论理想状况。在真实的实践中，我们通常发现很多公司在每个评审类型中按其喜好增添或者除去他们不喜欢的部分。我们通常也会听到公司谈论走查、同行评审，而他们使用的评审类型却并不符合 IEEE 1028 的要求。

由于基础大纲忽略了 IEEE 1028 标准的管理评审和审计功能，我们在此填补这一空白，从管理评审开始。管理评审的共同目标是监视进展、评估状态，以及对相关项目、系统、后续行为，或者过程制定决策（当然，在某些公司中，管理评审经常还用于不同的政治缘由，但是我们在这里可以忽略）。

被评审项所涉及的经理经常会执行这些管理评审。不同的利益相关者和决策制定者也可以支持他们。每个参与者的参与度可以彼此不同。

在某些情况下，公司会雇用外部顾问来进行评审。例如，RBCS 大部分的业务是为公司不同种类的测试过程和质量过程进行评估。这些评估是一个管理评审和审计的混合体，我们会在之后讨论。测试经理经常驱动这些测试评估，在这种情况下它们看起来更像一种管理评审。当外部的测试利益相关者驱动这些测试评估，这种评估更像是一次审计。

通常情况下，一个管理评审需要评审项目如何进行计划、估算和项目风险管理等。一个管理评审也要查看不同过程 and 控制的充分性。参与者必须准备这些评审，特别是需要提交状态信息的这些人。我们在对某些公司进行测试评估时，发现他们对测试小组的工作控制很弱以至于我们主要的建议就是，“迅速加入一些度量和追踪机制”。

通常来说，管理评审的结果包括行动事项、建议事项和待解决事项等。决策应当被记录在文档中，行动事项和建议事项的执行应当定期地检查。不幸的是，公司不总是遵循这些行动事项和建议事项。

让我们来看审计，这时候就变得非常的正式，而且在某些情况下，非常的具有对抗性。在一次审计中，人们非常有可能被合同、标准或行业最佳实践等所衡量；这也可能激发抵触情绪。高级大纲中说到审计是发现缺陷的最没有效率的评审方法，但是也要看具体的审计行为和受审计的公司。当我们的客户进行测试审计时，我们非常善于发现缺陷，包括项目缺陷和过程缺陷。

审计的一个基本要素是独立评估。就像一次管理评审，我们可以衡量一个过程、一个项目、一个后续的行为或者一个系统。但是，审计的另一个基本要素为是否遵守规则。审计可以由一个主要审计员和一个审计团队执行，或者由单个审计员执行。审计员通过进行面谈、亲自参与、检查文档和分析度量等来搜集证据。我发现面谈很有意思，特别是当一个审计对某些参与者有重大的利益关系的时候。在这种情况下，他们会试图误导或搪塞审计人员。



就像管理评审一样，审计的结果包括行动事项、建议事项和待解决的问题等。但是，它也包括对服从与否进行的评估。这种情况通常需要对多维区域或者目录进行衡量，就算 99 项是符合标准了，但是有一项不符合，那么该公司也通不过审计。如果被发现有不符合标准的事情，那么这个审计发现通常会加入失败项的补救措施。基于审计结果，公司应当将决策写进文档。它应当定期地检查行动事项、建议事项、纠错行为的执行，以及定期地进行标准符合情况的重评估。规范的或者合法的审计人员执行的审计，通常会有一个很好的结果，但是如果公司在在一个不规范的行业的话，它们通常不会完全遵循所有的行动事项和建议事项。

### ISTQB 术语

**审计：**对软件产品或者过程进行的一个独立评估以确定是否符合标准、指导方针、规范说明，以及基于客观标准的过程，包括使用文档详细列举出（1）待生产产品的形式或者内容，（2）产品被生产出来的过程，以及（3）衡量是否合乎标准和指导方针。

**审查：**一种依赖于对文档可视化检查的同行评审，用于检测缺陷；例如，违背开发标准以及不遵守概要级别文档说明。最正式的评审技术通常都是基于一个文档化的过程。

**管理评审：**对软件采购、供应、开发、运营，或者维护过程的一个系统性评估，通常包括监视过程、决定计划和时间表的状态，确认需求和他们的系统分配，或者评估管理方法的有效性达到合适的目标。

**技术评审：**一种同组讨论行为，关注于如何获得技术方法的合规性。

## 6.4 引入评审

### 学习目标：

- （K2）比较各评审种类的区别，揭示它们的相对长处和短处，以及各自适用的领域。
- （K3）在一个正式评审中领导一个评审团队遵循确定的步骤进行工作。
- （K4）列出一个评审计划作为项目质量测试计划的一部分，包括评审技术、发现的缺陷、员工的可用技能，以及和合适的动态测试方法的联合。

以下这些步骤在成功引入评审时非常有用：

- 保证管理支持：从预算角度看评审并不昂贵，就像测试自动化，但是它们需要充足的时间，特别是时间非常紧的情况下。
- 引导经理：你需要就商业用例的评审，包括成本、益处和潜在问题与经理进行一次诚实的对话。避免夸大其词。
- 组织合适的结构：保证你使用的不同种类的评审有文档化的评审过程。保证有可用的模板和规范。建立一个类似评审度量元数据库的基础结构，保证人们理解度

量元的使用而且他们会帮助你收集数据。如果你要进行地理上的分布式评审,确保你有合适的工具进行该过程。

- 培训:对参与者进行评审技术和过程的教育。
- 获取参与者支持:保证进行评审的人和被评审人感觉舒适。
- 做一些实验评审:试着犯一些错误,然后从中进行学习。
- 展示益处:你有一个定义好的商业用例,对不对?现在告诉管理层你已经完成你之前承诺实现的事项。当然你需要度量元来执行这一动作。
- 将评审应用到所有(至少大多数)文档:需求、合同、项目计划、测试计划、质量风险分析,以及相似的高可见性文档是明显的目标。但是,我发现针对缺陷报告的非正式评审有非常大的价值。

你不必在每个公司落实每一个步骤,而且不必保证这些步骤具有非常完美的连续性,但是如果你想要省略每一个步骤之前必须仔细地想这样做是否可行。

公司会在评审中投资必要的时间和金钱。经理们期望该投资会有回报。作为一个测试经理,你应当可以展示这种回报。示范评审投资的回报的概念和示范总体测试投资的回报很相似,关于总体测试投资回报,在第3章已经介绍过。

为了显示在评审投资上的回报,你可以使用的度量元包括通过修复缺陷或者处理失败而节省的成本。在系统测试中的缺陷成本是什么呢?如果在发行后,成本又会是多少?一张简单的表格可以显示评审的益处以及衡量评审完成后的成功与否。

不要忘记度量节省的时间回报。金钱不可能一直是经理们最大的问题。事实上,交付到市场的时间是比金钱更大的问题。所以如果你可以记录下一个缺陷在评审中发现时需要5个小时去解决,但是在系统测试中发现时需要25个小时去解决,这样你就会有一个坚实的商业用例来说明,在项目早期阶段对评审投资的时间是如何减少项目后期推迟交付的可能性。

另外,建立合适的度量元对于后续的监视是非常重要的。评审过程很容易只是作为一种仪式,于是价值就会下降。如果你看到评审的价值下降了,问一下你自己为什么。事实上,好处应当不断地增加。你应当寻找基于度量元的、可度量的方法来改进评审过程。确保你和你的经理及参与者将评审,以及评审过程的改进当作一项长期的投资。

### 6.4.1 评审中缺陷消除的有效性

本章前面我提到了评审类型、不同的正式性以及这些区别如何影响缺陷消除的有效性。我特别地引用了Capers Jones的一些数据。我们来看一下隐藏在这些数据之下的信息,Jones对成千上万个客户和项目进行了研究,在评审应用系统,以及评审不同种类的有效性中,他发现了很多有趣的数据。Jones提到非正式评审是效率最低的,具有部分正式元素的评审效率在平均水平,效率最高的是高正式性的审查。当然,为了让评审在每个正式级别都有高效率,你需要谨慎处理并对此过程进行公司层面的支持。<sup>2</sup>

2 表1中的数据摘自Capers Jones的“Software Assessments”、“Benchmarks”和“Best Practices”一书。



表 6-1 显示了 Jones 所观察的不同评审类型的缺陷消除有效性范围。缺陷消除有效性是在进行某些静态或者动态测试活动时已经存在的缺陷数和被成功发现并消除的缺陷数的百分比。Jones 基于两个因素对评审进行分析。第一个因素是工作产品被评审的种类。第二个因素是用于评审的成熟度级别。如果你将评审作为一种过滤器，数字能告诉你评审会是多么的有效。

表 6-1 评审种类和有效性

	至 少	平 均	至 多
需求评审	20%	30%	50%
概要设计评审	30%	40%	60%
功能性设计评审	30%	45%	65%
详细设计评审	35%	55%	75%
代码评审	35%	60%	85%

首先，想象你开始的时候有 1 000 个缺陷。你在评审中遵循最差经验，但是你至少对所有种类的项进行了评审。在这种情况下，你进入测试时还残留 166 个缺陷。现在，想象你开始的时候还是有 1 000 个缺陷。但是，这次你遵循最佳经验（也就是对所有种类的项进行了评审）。这次，你进入测试时还残留 3 个缺陷。

6.4.2 两个评审实例研究

下面是我们一个客户所使用的正式地变更管理过程的一个大纲。这些控制委员会会议就是正式的管理评审会议。

变更控制委员会过程

版本 1.0

1. 范围：变更控制委员会旨在帮助 Netpliance 首次发布的变更请求以及已经发布的产品的生产变更的处理。

2. 提交变更请求：变更请求应当通过开发或项目管理提交。一个变更请求必须是书面的而且至少应当包括以下一些准则：

a. 变更的定义（变更是什么以及为什么该变更是需要的）

b. 影响的部分（包括但不限于设备硬件、客户端软件、服务器团队、网络团队、标识、制造过程）

c. 需更新的文档（包括但不限于记录文档的计划、工程规范说明、用户界面说明和用户指南）

d. 变更的级别

紧急——需要马上注意。必须有相应的处理计划

正常——正常的处理

日常——将其作为下一次正常变更的一部分

e. 依赖性（与其他特定的变更、特定的决策和假设的依赖关系）

f. 如果采用的话，预计对组织金钱上的影响（节省或者花费）

g. 要求完成的日期

3. CCB 评审：

3.1 团队成员：配置管理应当请求相关人员的批准。正常的变更请求会在例行安排的 CCB 会议上依次进行评审。紧急的请求需要马上亲自提交或者发邮件给适当的部门进行批

准审核。需要的签名应该基于 1b 所定义的具体问题具体分析。所需签名的指导方针如下:

a. 开发

- 仅涉及设备 (Qadeer、Craig 或 Ken)
- 仅涉及客户端 (Wayne、Craig 或 Ken)
- 仅涉及设备和客户端 (Craig 或 Ken)
- 仅涉及服务器 (David G. 或 Ken)
- 涉及服务器、设备和客户端 (Ken)

b. 测试 (Rex 或 Greg)

c. 支持 (Chris 或 Tom)

d. 网络运营 (Steve)

e. 供应链 (Maeve 或 Greg)

f. 项目管理/发布管理 (Albert 或 Greg)

g. 财务 (Brandon 或 David H.)

h. 市场 (Munira 或 Annie)

4. 签名准则:

以下列举了所有变更的签名准则 (包括首次发布)

- a. 设备团队——签名意味着该次变更对于设备的功能没有影响或者对设备的影响已经完全测试过了, 而且对顾客没有大的影响。
- b. 客户团队——签名意味着该次变更对于软件的功能没有影响或者对软件的影响已经完全测试过了, 而且对顾客没有大的影响。
- c. 服务器团队——签名意味着该次变更对于服务器环境的功能没有影响或者对服务器环境的影响已经完全测试过了, 而且对顾客没有大的影响。
- d. 测试团队——签名意味着该次变更已经成为整个测试的主题, 测试结果已经提交给整个团队。
- e. 支持团队——签名意味着该次变更没有引入任何的影响客户的问题或者任何引入的问题已经有一个文档化的客户应对方案并已记录进支持数据库。
- f. 网络运营——签名意味着网络运营的基础已经为这次变更准备好了。
- g. 供应链——签名意味着供应商可以支持该次变更而且材料价目清单已经相应地更新和上传。
- h. 发行管理——签名意味着 CCB 中所有该次变更相关的签名已经收齐。也意味着所有必需的附件已经放好并受相应的文档控制。也意味着审批通过的 ECO 将被通知到所有的相关部门实施。
- i. 财务——签名意味着该次变更对财务的影响是可以接受的。
- j. 市场——签名意味着该次变更对于市场是需要的。

5. 附件要求:

- a. 设备变更——规格说明、图表、测试计划、测试结果。
- b. 客户端——发布说明、tar 文件、测试计划、测试结果。
- c. 服务器变更——发布说明。
- d. 文档说明/打包变更——改进的图表和文档说明。

6. 批准了? 变更控制委员会的每个成员, 就像第二部分和 1b 项所要求的那样, 需要对每次变更请求给予审批。批准应当包括一个批准日期, 表示批准者代表的职能区域从这个日期起已经准备好了。一旦所有的批准纳入配置管理, 所有相关方会收到一个通知。如果一个或者多个批准人没有批准这个特定的变更, 发起人会收到通知而且本次变更会被舍弃。

7. 实现变更: 正如在批准周期中承诺的一样, 所有受影响的团队负责实现该次变更。配置经理负责将所有变更通知到相应的团队并收集必要的相关文档。



让我来点评一下上述文档的一些关键部分。

首先，注意我们有一个定义良好的范围。变更控制委员会旨在帮助首次发行时的变更请求以及后续的对已发布产品的生产变更的处理。

还有一个要求，在开会前，提出变更建议的人必须为本次变更做好充分的准备。关于如何提交变更请求是有一定规则的。这些规则包括提交者应当提交的内容以及提交的途径。

接着这个文档定义了 CCB 应当如何评审这些请求。从定义 CCB 的团队成员开始，这个团队包括整个项目团队和所有相关利益者。

每一个成员组扮演一个定义的角色。子章节 3.2，“签名准则”，描述了那个规则。该文档也定义了当一个团队成员签署一项变更请求时意味着什么。签署一项变更请求不只是指出该次变更被通过了。在某些情况下，这个签名还意味着，“是的，我理解而且会支持这个变更，但是我不认为这是个好主意。”测试团队的签名意味着我们会测试或者已经测试了该变更但是不代表我们认为测试结果支持这次发布。

除了变更请求本身，在很多情况下，还需要许多额外的附件。这样文档才被认为是完整的。

在第四部分，“批准？”该文档讨论了变更请求的处理细节。最后，第五部分，“实现变更”，描述了一旦变更被批准并进入生产时，它要求每个团队必须尽到自己的责任。

从图 6-2 中，你可以发现对于我多次提到的因特网相关项目的客户端而言，市场需求文档中一系列模棱两可的项。这个图显示了一个 6 页列表的一部分。每个页面形容了我们在对该文档进行评审时发现的十几个模糊项。

邮件

1. 当用户 30 天的试用期结束后，邮箱就自动失效，他们会得到警告吗？

不是一个测试问题，因为用户需要打电话来改变服务级别。电话营销可以给予警告。[关闭]

2. 如果一个账户的级别从高级降为一般级别，然后决定恢复到高级，子账户的邮箱可以被恢复好吗？[待定]

3. 电子邮件从网页的接入是否是一个需求？之前的回复提到这不在 R1 范围内，但是后续的邮件指出该问题仍需要考虑。[待定]

图 6-2 实例：在 MRD 评审中发现的缺陷

### 6.4.3 引入评审练习 1

我们回头再来看一下第一次迭代开始之前的 HELLOCARMS 项目。假设由于你对质量保证具有专业技能，管理层委任你进行 HELLOCARMS 系统需求文档的评审策划工作。

确认以下项：

- 你要采用哪种评审技术？为什么？
- 你要采用哪种正式级别？为什么？
- 你期望找到哪些缺陷？
- 在评审团队中你会需要哪些技能？

如果是在教室里，那么最好分成3~5个小组进行讨论。如果你独立学习，需要自己完成。如果是在教室里，每个小组完成之后，对结果进行讨论。

建议用时15分钟来完成这个练习，包括讨论时间。

#### 6.4.4 引入评审练习1的参考答案

以下是我对上述4个问题的答案。

您要采用哪种评审技术？为什么？

按照我的经验，对于像这类项目的文档，我在执行评审时会使用走查。我觉得这是最适用的评审形式，参加评审的人员收到的制约比较少，效率自然就高。假如您邀请了合适的人，那么使用走查在建立统一和教育项目团队上会有很大的作用。但是，因为走查没有审查那么高的效率，我也会使用静态分析和质量风险分析（基于需求）来检测和消除额外的问题。

您要采用哪种正式级别？为什么？

除非Globobank项目团队有较高级别的正式评审的经验，我会保持使用低级别的评审正式性，以及依靠在之前章节中讨论的多过滤方法来保证这个关键文档的较高缺陷消除率。

您期望找到哪些缺陷？

我希望发现大量模糊的、矛盾的以及不可测的需求。因为只是快速地读一下文档就能发现这些问题。如果我在评审中发现这些问题而且有不相称数量的语法、拼写、格式以及其他表面错误，我会试着首先清除这些文档错误之后再安排一次新的评审。

在评审团队中您会需要哪些技能？

基本上，参加质量风险分析的同一个跨功能性团队也应当参加该项工作。

#### 6.4.5 引入评审练习2

如果是在教室里，那么最好分成3~5个小组进行讨论。如果您是独立学习，需要自己完成。选择一个评审类型然后将该评审类型运用到HELLOCARMS系统需求文档，侧重于重要的问题而不是表面问题。

如果是在教室里，每个小组完成之后，对结果进行讨论。

建议用30分钟来完成这个练习，包括讨论时间。

#### 6.4.6 引入评审练习2的参考答案

RBCS的高级会员Jose Mata对HELLOCARMS系统需求文档进行了评审。他使用Karl Wiegers在高级测试分析课程中介绍的检查列表之后提供了以下反馈。<sup>3</sup>

- 内部的相互参照正确吗？如果我们在另一个文档或者同一个文档中引入某项，那么这个引用是有效的吗？
  - 不是。010-010-040表明：“域验证细节是在一个独立文档中描述的”，但是在

3 你可以在Karl Wiegers的“Software Requirements”一书中发现这个检查列表。



需求文档中没有定义该文档。

■ 细节级别是连续和合适的吗？

- 不是。将 010-010-180 作为例子：“为 Globobank 零售分行提供特性和屏幕以支持其运营。”该事项描述太模糊以至于很难顺利进行。
- 010-010-170 表明：“通过因特网支持应用的提交，也包括保证未受训的用户合理运用该应用的能力。”这是一个巨大的模糊的需求。
- 010-010-190 到 010-010-240 开始于“支持市场、销售，以及某些过程，”这样太模糊以至忽略了重要功能。

■ 需求有没有为设计提供一个充分的基础？

- 没有，010-010-070 表明：“询问每个申请是否和 Globobank 存在关系；例如，任何活期或者定期账户，”但是该表并不完整，而它应该是完整的。
- 010-010-080 表明：“从开始到拒绝、谢绝、或者接受就维持应用的状态……”，但是我们不知道这些状态是否是子类或者它们是否是综合性的。
- 010-010-150 表明：“对有 Globobank 银行运营的所有州、省和国家提供内包和外包的电话营销支持”，但是该列表没有很好地被定义。
- 010-010-160 表明：“通过提供具体的合作伙伴屏幕、标志、接口和品牌来支持代理人或者其他商业合作伙伴”，到目前为止，屏幕或者接口区域还没有定义。
- 010-010-250 表明：“支持灵活的定价模式包括引入定价、短期定价，以及其他”，但是“其他”需要预先定义。
- 每个需求是否包括了优先级？
- 是的。

■ 是否所有的外部接口已定义？

- 不是的。我们不知道这个信息有多完备。有内在的数据结构，但是没有显式地定义。内含的接口如下：
  - 贷款文件打印系统：010-010-050, 010-010-100, 010-020-050, 010-030-040, 010-030-060, 010-030-070, 010-030-080, 010-030-120, 010-030-130, 010-030-140, 010-030-145
  - GLADS: 010-010-070
  - 主机评分：010-020-020, 010-030-020
  - GloboRainBQW: 010-030-010

■ 还有遗漏的信息吗？如果有，是否已经标记为 TBD（“待定”）？

- 是的。

■ 期望行为是否已存档？

- 没有。例如，010-010-080 表明：“从开始到拒绝的过程维持应用的状态……”，但是如何以及在哪里维护状态还没有开始。

■ 是否每个需求是明确、准确，以及不模糊？

- 不是。例如，010-010-070 表明：“询问每个申请是否和 GloboBank 有关系”，

但是申请是怎么请求的还不清楚。

- 010-010-100, 010-030-040 和 010-030-070 表明:“允许用户在一个独立的屏幕向客户展示已有的债务,如果顾客将退休的话……”,但是该屏幕如何独立并没有定义清楚。
- 010-040-010 表明:“支持一致同意的安全需求(加密、防火墙等)”,这个也很模糊。
- 010-040-060 表明:“对所有金融应用进行欺诈侦测”。这个陈述很模糊,尤其是对于优先级为1的需求。
- 是否每个需求都可以被验证?你可否设计一个测试来验证需求是否达到?
  - 不是。010-010-150 表明:“提供内部外部电话营销支持……”,这个比较模糊因此不可被验证。
  - 010-010-190 到 010-010-240 开始于“支持市场、销售,以及处理……”,市场部分不可被验证。
  - 010-010-250 表明:“支持灵活的定价模式,包括引入定价、短期定价,以及其他”,但是“其他”不可被验证。
  - 010-030-150 表明:“支持电脑电话集成来为内部的电话营销,品牌商业伙伴提供定制的市场和销售支持”,该陈述也很模糊而且不能被验证。
- 每个需求都在范围之内吗?
  - 不是。例如,010-010-170 表明:“通过因特网支持提交应用,也包括未受训的用户正确地使用应用的能力”。这个超出 003 节的范围,因为允许基于因特网的客户进入对于后续发布很困难。
  - 010-040-030 表明:“允许外包电话营销员查看信用等级但是不允许他们查看申请者的实际信用分数”。这个超出 003 节的范围。
  - 010-040-050 表明:“允许因特网用户浏览潜在的贷款但不泄露用户信息……”。这个超出 003 节的范围。
- 是否每个需求都没有内容和语法错误?
  - 是的。
- 需求可以在限制内被实现吗?
  - 可能不行。010-040-060 表明:“支持对所有金融应用进行欺诈侦测”。这个不太可能被实现。需要定义特定的检查。
- 是否所有的安全性考虑被合适地定义?
  - 不是。特定种类的用户,以及他们的许可权限没有被提出。用户名字和密码没有提出。特定数据的加密没有提出。维护和清洗需求没有提出。服务器物理安全需求没有提出。
- 是否每个需求被唯一和正确地识别?需求的间隔尺度是否可能从测试到需求都被追踪?
  - 不是。例如,010-010-190 到 010-010-240 开始于“支持市场、销售,以及处理……”。需求的间隔尺度太大了。



- 010-010-250 表明：“支持灵活的定价模式包括引入定价、短期定价，以及其他”。这个和其他的复合需求，如果它们是分隔的有序需求就要更加清晰。这个可能是会有重复，但是需求应当更清晰而且在开发和测试工作的范围衡量中应该要有更多的平衡。
- 我们是否停留在合适的需求范围之内，而不是解决范围？换句话说，是否所有的需求都是实际需求而且不是设计或者实现的解决方案？
  - 不是，010-010-100，010-030-040 和 010-030-070 表明：“允许用户在一个独立的屏幕中向客户展示已有的债务，如果顾客将退休的话……”。识别一个独立的屏幕看起来是一个设计的细节。

## 6.5 评审的成功因素

### 学习目标：

(K2) 解释在进行评审中不考虑技术、公司因素和人员因素所造成的风险。

有很多因素会影响评审的成功或者失败。大纲将这些因素分为 3 类：技术因素、公司因素，以及人员因素。我们先讲技术因素。

首先，确保你正确地遵循预定义的过程。这对于像审查这样的正式评审种类来说特别地棘手。但并不意味着你不能调整这些过程，通常在一开始就进行控制是个好办法。<sup>4</sup>

现在，我来谈谈商业用例的问题。为了支持你的商业用例，你不得不记录下评审的成本（按照工作量）以及公司获得的好处。评审的问题是其所获得的好处往往是在公司付出成本之后才显现。这点适用于所有测试，尤其对评审而言，特别是如果你忘记对价值进行评估。

记住你不必要等到一个文档结束之后才开始评审它。在处理一些关键事项时，你可以而且应当评审早期的草稿或者部分文档。这有助于在将其加入整个文档前识别并防止各种缺陷。

也就是说，确保待评审事项的意义具有特定的规则。对还不成熟的材料进行评审是在浪费大家的时间。如果你对还在改进过程中的材料进行评审，则会使评审人员失望并浪费时间。人们对在某些活动上浪费时间感到非常沮丧，并且会导致其放弃这种行为，也就是说评审过程会慢慢消失。所以要有一些入口准则，其中应当包含一个简单规则，那就是每个人都要出席而且都要准备好。

对于评审来说检查列表是很有用的。没有检查列表很容易遗漏重要的事项。所以一定要有些检查列表。你可以从有名望的行业专家那里获得检查列表模板，但是务必保证将这些检查列表修改成适合你公司的样式。这些检查列表应当阐述你所发现的共同缺陷。而且，对于不同种类的文档要有不同的检查列表，比如说需求、用户用例以及设计。最后，对不同的评审过程要有不同的检查列表。

4 对于审查有一本很好的参考书，请参见 Tom Gilb 和 Dorothy Graham 的“Software Inspection”一书。

不同的应用适用不同的正式性级别。要做好准备使用超过一种的评审类型，考虑你的目标。在发给用户前是不是要做一个快速的文档清理呢？为了提高一些技术设计决策？为了教育利益相关者？为了生成管理信息？

我提到过关于“两双眼”的规则，我尽量不去违反这条规则。但有时候，截止日期比较紧，做不到履行“两双眼”规则。但是，你应当评审，或者更好些，审查所有特别重要的文档。如果需要做出一个重要的决策，比如是否要签合同，那么确保之前审查了提案、合同，或者概要需求规格说明。如果预期有一个大的花费，应进行一次管理层评审来通过该预算。

对于大型文档来说，你可以使用有限子集的抽样来估计整个文档中的缺陷数量。基于你对总计缺陷数量的估计，你可以决定是否需要一次新的评审。记住这个取样方法不适用于文档清理或者编辑工作。

注意不要转移注意力。很容易就会发现不少小的格式、拼写和语法错误。要侧重于内容而不是格式才可能找出最重要的缺陷。

最后，正如我之前所说，通过使用你学到的知识来持续地改进评审过程。

现在，还有一些公司因素。首先，保证你和项目团队中的其他经理会计划和估计充分的时间，尤其是有截止日期压力的情况下。在软件过程早期忽略高效率的缺陷消除行为，虽然这样做可以加快结束时间，但并不是一个正确的行为。可惜这种做法在软件工程中非常得普遍。

你需要度量元来度量评审的好处，但是对度量元要小心。有件事需要记住，投入人力到某些评审会发现很多缺陷。而有些评审不会。有一些数学模型用于预测缺陷密集度，但是它们不在本书讲述的范围内。注意不要使用简单模型。最重要的，绝不要将评审缺陷度量元用于个人能力评估。由此引起的个体的反感会毁了整个过程。

保证缺陷的修复有足够时间。假设测试行为不会发现任何缺陷，这是一个最经典的测试的错误实例。

保证过程涉及正确的参与者。由摩托罗拉在 2000 年中做出的研究表明，正确的参与者是评审成功最重要的标志。关键是评审要加入技术人员或者学科问题专家。有正确的参与者意味着保持平衡，保证评审团队中有来自所有主要团队的代表人。而且，有正确的参与者意味着参与者要理解评审过程，通常是对参与者进行正式评审类型的教育。第二重要的标志，正如该研究发现的，是保证有正确数量的参与者，因此要仔细地想一下谁以及合适的人数。<sup>5</sup>

如果你在一个中到大型的公司使用评审，那最好举办一个评审论坛允许人们分享各自的经验和想法。这个论坛的相关信息可以被主持人或者主管保留。

评审中与会者如果没有贡献就没有任何意义，确保他们事先都进行了准备，另一点就是与会者有时会比较沉默，但并不代表他们没有好的观点。

在处理关键文档时，使用最强大，最正式的技术。记住 Jones 在评审有效性上的数

---

<sup>5</sup> Jeff Holmes, "Identifying Code-Inspection Improvements Using Statistical Black Belt Techniques", Software Quality Professional, 2003 年。



字。在每种文档中你可以留下百分之多少的缺陷？

保证你有一个合适的过程来评审过程改进。如果没有度量元的支持，你很可能会走错方向。确保对过程改进进行评审包括辨别及庆祝所取得的改进。

最后，是人员因素。首先，和其他经理一样，给所有的利益相关者和参与者打预防针，告诉他们会有缺陷。确保评审过程中他们的出现不会让人感到意外。确保他们有返工和重复评审的时间。在当今工作中，人们趋向于过高地估计自己。如果这样，就会在文档中遇到很多问题，这可能是一个痛苦的经历，因为这将意味着延时。

评审领导人不是 Torquemada，西班牙第一位宗教裁判所大法官，非常地残暴。拷问台、断头台和水刑等都不是评审的工具或者技术。评审过程对作者应该是一次积极的体验，他们可以从中，也就是他们所尊敬的同行中学习如何将他们的工作做得更好。我依旧记得，在刚开始我的测试生涯时，两三个有经验的师傅参与的评审给了我很大的帮助。如果作者有比较差的评审体验，小心别强迫他们服从某个评审。最好由管理层来处理这件事情。

基于评审在缺陷定位和缺陷消除方面的高效率，我们应当为发现缺陷感到高兴，而不是感到沮丧。确保让人们明白这是节省时间和金钱的好机会。不要试着在发现缺陷的时候推卸责任或者对别人强加指责。

要观察人们的对话。我们需要建设性的、有帮助的、体贴的，以及客观的讨论。确保人们会思考深层次的问题，包括在评审中如何才能更符合项目的要求。

### 6.5.1 回顾一个早期用例研究

在本章前面部分，对因特网相关项目的客户端市场需求文档的评审中，出现了很多问题。有意思的是，在该文档的评审中发现的问题，测试团队没有在开发之前或者开发过程中解决。当然，我猜想你可能会说，“Rex，最终一些程序员做了决策然后解决了这个问题。”

我很同意这个观点——最终，系统对于每个给予的状况做了些事情，这些事情也确实是程序员要他们做的。但是，我认为这不是解决严重问题的好办法。

接下去发生的就是开发和市场部门决定用公开问题进行处理。我们被告知不要用这些事情来烦他们，他们让我们通过开发并最终运行测试来进行处理。我们估计了在测试设计和开发的具体程度，我们需要达到哪种程度，以及不借助额外信息到达那里会有多难。我们推断会因为缺乏清晰定义的产品需求而导致 20%~25% 的测试效率丢失。对于这件事情，我们的客户联系人值得回味地跟我说，“我宁愿付给 RBCS 额外的钱也不愿意教这些人如何写需求。”

我笑着回答，“是的，只要你的支票不拒付，这是你做出的最好选择。”

在测试执行中，无效率意味着会有高比率的假正面和假负面现象。它也减少了测试覆盖率，由于我们不得不在测试执行中花费时间记录系统实际行为，我们期望的行为，以及这两者之间的区别。所以，在这里有一个经典的实例，解决在截止日期压力下不能很好计划和估计充分的评审时间。

## 6.5.2 评审的 IEEE 1028 标准

让我们来看一下 IEEE 1028 中的评审标准,如图 6-3 所示,介绍了基础大纲。该标准的第一个部分是一个概论。它包括了标准的目的、覆盖的范围和 IEEE 1028 标准符合的指导方针、该标准的机构以及如何在公司中应用该标准。

1. 概论
  - 目的、范围、一致、组织、应用
2. 参考
3. 定义
4. 管理评审
  - 职责, 输入/输出, 入口/出口准则, 步骤
5. 技术评审
  - 职责, 输入/输出, 入口/出口准则, 步骤
6. 审查
  - 职责, 输入/输出, 入口/出口准则, 步骤, 数据收集, 过程改进
7. 走查
  - 职责, 输入/输出, 入口/出口准则, 步骤, 数据收集, 过程改进
8. 审计
  - 职责, 输入/输出, 入口/出口准则, 步骤

图 6-3 软件评审的 IEEE 1028 标准

第二部分是“参考”,正如你所想的,引用了其他文档、标准等。第三部分,“定义”,定义了标准中使用的术语。

第四部分讲述管理评审。管理评审超出了基础级别的范围。在该部分中,IEEE 1028 标准讨论了谁在管理评审中有什么责任,管理评审的输入和输出,开始这种评审的入口准则和辨别结束的出口准则,以及一个管理评审应当遵守的过程。

第五部分讲述了技术评审。在这个部分中,IEEE 1028 标准讨论了谁在技术评审中负有什么责任,技术评审的输入和输出,开始这种评审的入口准则和辨别结束的出口准则,以及一个技术评审应当遵守的过程。

第六部分讲述了审查。就跟前两个部分一样,这个标准讨论了谁在审查中有什么责任,审查的输入和输出,开始这种审查的入口准则和辨别结束的出口准则,以及一个审查应当遵守的过程。但是,因为审查比技术评审更加正式,这个标准也讨论了从评审过程中收集数据并对评审过程实施改进。

第七部分讲述了走查。就跟其他部分一样,这个标准讨论了谁在这个走查中有什么责任,走查的输入和输出,开始一个走查的入口准则以及辨别结束的出口准则,以及一个走查应当遵守的过程。由于走查和审查有相同的正式性级别,这个标准也讨论了从评审过程中收集数据并对评审过程实施改进。

最后,第八部分讲述了审计。就跟其他部分一样,该标准谈论了谁在这个审计中负有什么责任、审计的输入和输出、开始审计的入口准则辨别结束的出口准则,以及审计应当遵循的过程。



## 6.6 认证考试模拟题

在每个章节的最后，我们会准备一个或多个考试问题实例来加强你对该章内容的理解并用来准备将来的 ISTQB 高级测试经理考试。

1. 以下哪个缺陷类型在评审中比在动态测试中更容易被检测到？
  - A 回归
  - B 可维护性
  - C 性能
  - D 可靠性
2. 以下哪种评审类型可以在现行待评审项中检测到缺陷的最大百分比？
  - A 非正式
  - B 静态分析
  - C 模拟
  - D 审查
3. 假设你是某项目中的一名测试经理，该项目创建一个可编程恒温器用来控制家用的中央供热系统、空气流通设备和空调系统。除了正常的 HVAC 控制功能，这个温度控制器还能够将数据下载并输入到在个人电脑中运行的一个基于浏览器的应用，并将此用于后续的分析。

在质量风险分析中，你发现基于浏览器的应用和不同的 PC 配置之间存在兼容性问题，这非常有可能使该应用成为一个质量风险项。

选择以下在质量和测试计划中应当发生的所有的动作来保证项目团队会减少在发布中的风险？

  - A 仔细地选择支持配置的列表
  - B 和项目利益相关者评审支持配置的列表
  - C 基于技术属性仅支持单一浏览器
  - D 基于可能的测试资源降级风险可能性
  - E 在测试执行早期测试支持的 PC 配置
4. 假设你是一名负责银行的集成测试、系统测试和验收测试的测试经理。你现在的项目用来将一个现有的自动化出纳机器系统升级使得该系统可以允许顾客从支持的信用卡获得现金预付款项。这个系统应当允许从\$20~\$500 的现金预付款项，并且支持所有信用卡。包括 American Express、Visa、JCB 卡、Eurocard 和 MasterCard。

银行让你负责组织一个需求规格说明的评审会。以下哪个是评审的风险项？

  - A 你不知道支持的信用卡
  - B 在评审中你没有包含合适的相关利益人
  - C 你不知道要阐明哪种测试级别
  - D 你没有收到需求规格说明

## 第7章

# 事件管理

根据目前的趋势来推断，2017 年 100%的发布预算会用在考虑变更上，而目前的预算是 0%。好处是这意味着测试版本发布将不花一分钱。

——V.V.L., 欧洲测试工程师。

高级大纲的第 7 章是关于事件管理。就像基础大纲解释的一样，任何时候，实际测试结果和期望测试结果有出入时就发生了一个事件。高级大纲使用 IEEE 1044 标准来聚焦于事件生命周期和测试人员为了事件报告所需收集的信息。高级大纲的第 7 章有 6 个部分：

1. 概述。
2. 何时可以发现一个缺陷。
3. 缺陷生命周期。
4. 缺陷域。
5. 度量元和事件管理。
6. 沟通事件。

下面依此来看一下每个部分以及它们和测试管理之间的联系。

## 7.1 概 述

### 学习目标:

能回想起该部分内容即可。

时间管理对于所有测试人员和测试经理来说都是一个基本技能。作为测试经理，我们非常关心这些过程。我们希望缺陷的处理过程是顺利且及时的，从识别到调查到改正到总结。测试人员最关心的是精确地记录事件并在该过程的总结部分执行合适的确认测试和回归测试。

测试人员按照他们的职责会有某个不同的侧重。测试分析人员按照业务和用户的需求比较实际和期望的行为。测试技术分析人员评估软件的行为和可能需要其对运用后续技术的深刻见解。



## 7.2 何时可以发现一个缺陷

### 学习目标:

能回想起该部分内容即可。

只要我们已有一个需求规格说明的初稿，就可以通过静态测试来检测缺陷。我们只要有可执行的单元就可以通过动态测试来检测失效，也就是缺陷的症状。

测试就是一个过滤行为，所以为了达到可能的最高质量，我们应当在软件生命周期中不断进行静态和动态测试行为。除了过滤缺陷，如果我们有很多早期的过滤，像需求评审、设计评审、代码评审、代码分析等，则会在早期发现缺陷并且消除它们，减少了总体成本，并保证了进度。

在动态测试中，当我们看到期望结果和实际结果不一致时，我们不当自动认为这个就是被测试系统的缺陷。也可能是测试本身存在缺陷。

#### ISTQB 术语

**缺陷 (bug):** 可能会导致软件组件或系统无法执行其定义的功能的瑕疵，例如：错误的语句或变量定义。如果在组件或系统运行中遇到缺陷，可能会导致运行的失败。

**错误:** 人为地产生不正确结果的行为。

**失效:** 组件/系统与预期的交付、服务或结果存在的偏差。

**事件:** 任何有必要调查的事情。

**事件日志:** 记录所发生的（例如，在测试过程中）事件的详细情况。

**根本原因分析:** 一种旨在识别缺陷根本原因的分析技术。通过纠正缺陷的根源，期望将缺陷再次发生的可能性降到最低。

## 7.3 缺陷生命周期

### 学习目标:

(K3) 遵循按照 IEEE 1044—1993 标准所建议的事件管理生命周期过程处理缺陷。

从图 7-1 中，可以看到 IEEE 1044 中的事件管理生命周期，包括一张 IEEE 1044 的配对图，表现了在一个事件追踪系统中的典型缺陷报告如何和这个生命周期相联系。下面来看一下这个过程。

我们假设所有的事件在它们生命周期会遵循某种状态顺序，从最初的识别到最后的总结。不是所有的事件都会经过相同的状态顺序，就像在图 7-1 中所看到的。IEEE 1044 缺陷生命周期由 4 个步骤组成：

1. 识别：当我们观察到一个异常（观察到的是一个事件）就要进行识别，也就是意

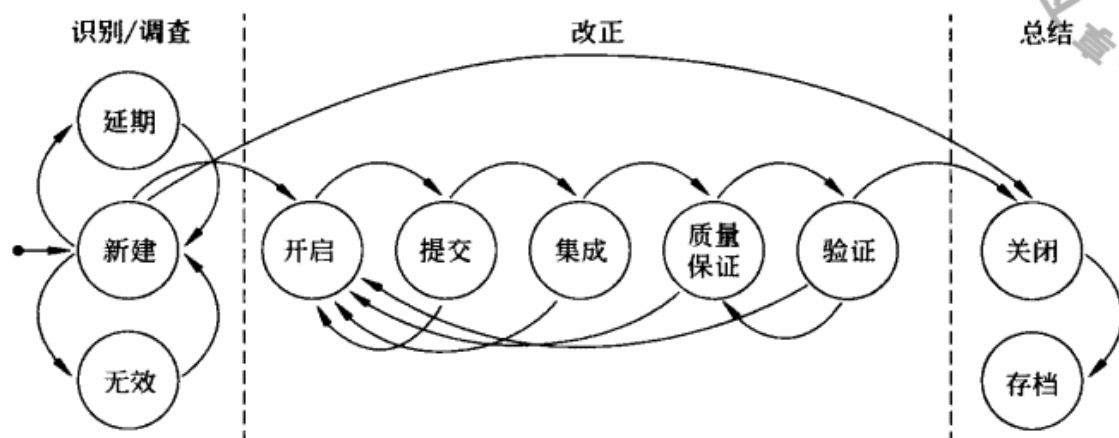


图 7-1 IEEE 1044，时间管理生命周期

意味着一个潜在的缺陷。识别会发生在软件生命周期的任何阶段。

2. 调查：在识别之后，对事件的调查就会发生。调查可以揭示相关的问题以及提出解决方案。一种解决办法是断定该缺陷不是由实际缺陷引起，例如，测试数据可能会有问题。

3. 改正：调查的结果会激发改正步骤。我们可能会决定解决这些缺陷。我们可能需要采取措施来预防将来类似的缺陷。如果缺陷被解决，就要进行确认测试和回归测试。任何被缺陷阻止的测试会重新进行。

4. 总结：根据推断出来的行动，这个事件到了总结阶段。我们这里主要对捕获后续信息以及将事件移到一个最终状态感兴趣。

让我们来看一下其是怎么在实际项目上工作的。在图 7-2 中，可以看到我们在 IVR 综合系统项目所使用的缺陷追踪系统的生命周期。我们用 Microsoft Access 为客户创建缺陷追踪应用。这个项目在像 Bugzilla 这样的开源软件系统被广泛应用之前产生，所以在当时是很节省成本的。

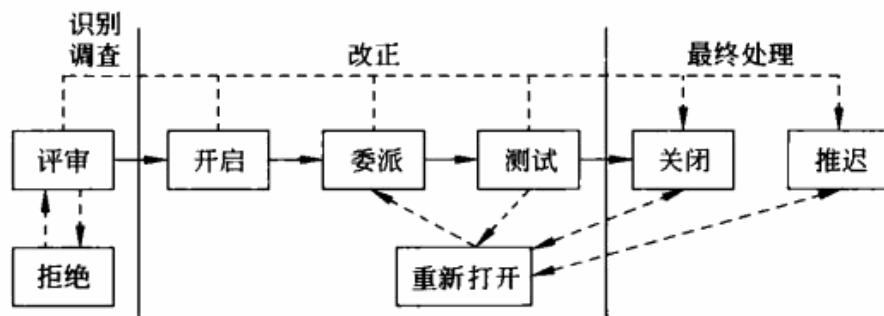


图 7-2 实例：缺陷生命周期

### ISTQB 词汇表

**异常：**任何和基于需求文档、设计文档、用户文档、标准或者个人的期望和预期之间偏差的情况，都可以称为异常。异常可以在但不限于下面的过程中识别：评审 (review)、测试分析 (test analysis)、编译 (compilation)、软件产品或应用文档的使用等。



如果回到图 7-1，可以看到状态和步骤配对非常相似，区别很小。一方面，我们认为推迟是一个最终的处理。我们没有一个无效的状态，但是如果我们决定永久地拒绝该状态，我们可以将一个报告以无效作为理由而进行关闭。拒绝也是一个最终的处理。

另外，我们使用单个联合指派任务状态而不是分开提交和构建状态，因为我们不会离开修复时间来追踪构建时间。我们使用单个联合测试状态而不是分隔 QA 和验证状态，因为我们的测试团队获得授权已决定该问题是否被解决。

最后，我们没有存档状态因为我们假定报告在它们的最终状态会无限期保持。

### 7.3.1 缺陷生命周期练习

假设将一个电话银行业务员团队参加 HELLOCARMS 测试作为一个 Beta 测试。银行业务员会输入客户实时申请，但是他们事后要捕获信息并输入系统来保证 HELLOCARMS 系统的缺陷不影响到顾客。

银行业务员不是受过培训的测试人员而且也不愿意花费时间来学习测试基础。所以，为了避免银行业务员输入的书面事件报告质量太差以及在事件报告度量元中引入争议，管理层已经决定当银行业务员发现问题时要通过电子邮件通知测试分析师输入报告。

以下是一个测试分析师收到的一份银行业务员描述问题的邮件：

我在为一个有很好信用度的客户输入数据申请房屋净值贷款。她拥有一栋高价值的别墅，而且贷款额度还不是很大。

在屏幕上，HELLOCARMS 弹出“升级到由电话银行客户经理处理”的信息。但是，我单击了“继续”按钮后，在我没有输入任何的电话银行客户经理授权码的情况下，它仍然允许我继续进行的操作。

从这之后的客户应用中，所有的行为都是正常的。

我有另一个客户有一个相似的情况——高价值的别墅，中等大小的贷款额度——在那天晚些时候打进来电话。同样的问题，我不用输入任何授权码就可以进行了。

回答以下 3 个问题：

1. 当测试分析师接收到报告的时候，这个报告属于 IEEE 1044 中的哪一步？
2. 为了让该报告进入 IEEE 1044 生命周期的下一个状态，测试分析师需要采取什么行动？
3. 在这些行动之后，你期望该报告能进入 IEEE 1044 生命周期的哪种状态？

答案请见后面的缺陷生命周期练习报告。

如果是在一个教室中学习，那么你可以选择在一个 3~5 个人的小团队中共同完成。

### 7.3.2 缺陷生命周期练习参考答案

以下是我对上面 3 个问题的答案。

当测试分析师接收到报告的时候，这个报告属于 IEEE 1044 中的哪一步？

它属于调查阶段，尽管电话银行业务员有可能没有收集识别阶段需要的所有信息。

为了让该报告进入 IEEE 1044 生命周期的下一个状态，测试分析师需要采取什么行动？

首先,评估每个相应的报告和识别其影响的类别和数据域以确定电子邮件或者其他文件是否提供了该信息。如果不是,获取可能的信息来完成识别步骤。

其次,评估每个相应的调查结果和结果中影响的类别和数据域,确定这个阶段还需要收集什么信息。

在这些行动之后,你期望该报告能进入 IEEE 1044 生命周期的哪种状态?

现在应该处于开启状态,准备分配给一个特定的开发人员。当然,缺陷分类修复委员会或者变更控制委员会可能认为这个不是缺陷,然后决定关闭缺陷——这有点奇怪,但这是可以预见的。同时,缺陷分类修复委员会和变更控制委员会可能决定这个行为不正确,但是对于现在来说是可以接受的,于是他们会决定将该缺陷报告定为推迟的状态。

### ISTQB 术语

配置控制委员会(也叫变更控制委员会或者缺陷修复协会或者事件修复协会):负责评估、批准或拒绝配置项修改的组织,此组织应确保被批准的配置修改的执行。

## 7.4 缺陷域

### 学习目标:

(K3) 运用 IEEE 1044—1933 标准以及适用的缺陷分类法评估缺陷报告以用于改进报告质量。

当然,驱使事件从一个状态到另一个,以及由此从生命周期的一个步骤到另一个的人,就是测试分析人员、程序员,以及其他项目参与者。他们根据掌握的事件情况将其一步一步移动到最终位置。因为缺陷要从一个所有者到另一个所有者,所以需要状态;一个缺陷追踪系统必须捕获该获知过程。因此,在每一步中——其实是包含在每个状态中——有 3 个信息捕获行为:

- 记录。
- 分类。
- 识别影响。

工作方式如表 7-1 所示。

在确认阶段,我们记录支持数据。我们基于所观察到的重要属性对报告进行分类。我们基于觉察到的影响来识别影响,可能该影响和最终影响评估并不一样。

在调查阶段,我们更新和记录更多的支持数据。我们基于调查中未被覆盖的属性的重要性对信息进行更新和分类。我们也会基于调查更新可能会产生的影响。

在改正步骤中,我们基于采取的行动记录新的支持数据。我们也基于采取的行动加入分类数据,我们也会基于行动更新影响。



表 7-1 IEEE 1044 分类过程

步 骤	活 动		
	记录……	分类……	识别……
1. 识别	包含支持数据	基于重要属性	基于发现的影响
2. 调查	更新以及加入支持数据	对重要属性更新和分类	基于调查进行更新
3. 改正	基于采取的行动加入数据	基于采取的动作加入数据	基于动作进行更新
4. 总结	基于总结加入数据	基于总结	基于总结进行更新

最后，在总结阶段，我们基于总结记录最终数据，基于总结调整和最终确定分类并获得最终影响评估。

注意我一直在讨论数据和分类。IEEE 1044 标准包括对每一步中的每一个行为进行强制性和选择性地支持数据及分类。我们会在之后进行评审。顺便说一句，当我说“强制性支持数据和分类”时，我是指符合 IEEE 1044 的强制标准。

每个数据项和分类都和一个步骤或者行为紧密联系。IEEE 1044 标准定义了两个字母的代码：RR（识别）、IV（调查）、AC（改正）、IM（识别影响）以及 DP（总结）。

我们之后会看一下这些数据项和分类。当我们做这些的时候，不要看见树木却忽略了森林。通常要问自己的重要事情不是“我的事件管理系统和 IEEE 1044 符合吗？”，而是“这个数据和分类方法对于捕获缺陷有用吗？”。

以下列表展示了识别步骤的分类标准：

- 项目行为（代码 RR1XX）：事件被发现的时候你在做什么？这是 IEEE 1044 标准强制需要遵守的一个项。
- 项目阶段（代码 RR2XX）：该项目处在 IEEE 1044 标准的哪个阶段？你不得不调整项目阶段使其适合你的生命周期。
- 可疑原因（代码 RR3XX）：你觉得哪些可能是原因（可选项）？我发现，在很多情况下，捕获这种数据可以帮助你的程序员修复缺陷，特别是如果你的测试人员技术水平很高。
- 可重复性（RR4XX）：你可以重现这个事件让它再发生一次吗？我认为 IEEE 将此项列为可选是有问题的，因为重现问题的信息是好的事件报告的基础之一。
- 症状（RR5XX）：该事件表明发生了什么现象（强制项）？
- 产品状态（RR6XX）：如果事件没有解决的话那么产品的可用度是多少（可选项）？我也不同意 IEEE 对该项标准列为可选项。

代码后面的 XX 字符表明它们是有层次结构的。在每个项中还有子分类。例如，每一个项目行为选项有一个具体的代码，就像 RR110、RR120 等。IEEE 1044 同时定义了这些选项，但是我们不会过于关注该标准的细节。

以下内容展示了识别步骤的数据：

- 当你看到这个事件的时候你工作的环境是什么样？你应当获得产品硬件、产品软件、数据库、测试支持软件、平台、固件，以及其他有用的信息。
- 你可以获得哪些源头的细节信息？你应当获得测试人员姓名、观察到事件的日

期、代码或者功能性区域，分配商（测试对象的版本是什么），以及联系信息像电子邮件地址、地址、电话号码，以及公司 ID 号。

- 你什么时候看到这个事件的？包括操作时间（例如，从上次重启或者正常运行时间）、墙上时钟时间、系统时间以及 CPU 时间。
- 供应商的信息是什么？包括公司、联系人姓名、供应商 ID 号、期望的决议以及期望解决的时间。

以下内容展示了调查步骤的分类：

- 实际起因（IV1XX）：什么真正导致了事件（强制性）？
- 起源（IV2XX）：事件的起源是什么（强制性）？内在的错误是什么？
- 类型（IV3XX）：什么类型的缺陷导致了失败（强制性）？这个是缺陷分类的问题。

记住你可以在这个步骤中更新之前步骤中的分类。

以下内容展示了调查步骤数据：

- 我们能获得什么确认信息？收到了什么数据，分配了什么报告号码，谁是调查者，调查计划开始和结束的时间，调查实际开始和结束的时间，花费了多少人小时，我们在哪天接收到确认，我们在调查中使用了哪些文档？
- 我们能获得什么验证信息？什么是事件发生的源泉，我们怎样验证确认过程中的数据？

记住你可以在该步骤中更新之前步骤中的分类。

以下内容展示了改正步骤的分类：

- 解决（AC1XX）：事件应当在何时以及如何解决（强制性）？
- 改正行为（AC2XX）：应当采取什么措施防止将来类似事件的发生（可选）？

记住你可以在这个步骤中更新之前步骤中的分类。

以下内容展示了改正步骤数据：

- 我们可以捕获哪些解决方案信息？哪些测试项需要修复，哪些该项中具体的组件需要修复，我们怎么用文字来形容这次修复。哪天是计划行动结束的日子，指派的人员是谁，哪天是修复计划结束的日子，或者，如果该修复是推迟状态的，我们的依据是什么？

### ISTQB 术语

**优先级：**赋予某项（业务）重要性的级别，如缺陷。

**严重性：**缺陷对组件/系统的开发或运行造成的影响程度。

- 我们能够捕获哪些解决行动的信息？该任务完成的日期是哪天？哪个组织被委派来验证该解决措施，哪个人被委任用于验证该解决措施？

记住你可以在这个步骤中更新之前步骤中的分类。

以下内容展示了总结步骤分类和支持数据：

- 处置（DP1XX）：问题最终是怎么解决的（强制性）？



- 我们应当捕获哪些事件异常的总结信息？实现了哪些行动，报告在哪天关闭，哪天文档完成更新？什么时候通知了客户，参考文档数量会有多少？
- 我们应当捕获什么验证信息：执行验证的人叫什么名字，在哪天进行了验证，验证了哪个版本，用哪些方法进行验证，使用哪些测试用例进行验证？

记住你可以在这个步骤中更新之前步骤中的分类和数据。

现在，通过这个生命周期，进行了影响分类的制定和修改。我们来看一下某些受影响的分

- 严重性 (IM1XX)：对于系统的影响是什么（强制性）？
- 优先级 (IM2XX)：事件的相对重要性是什么（可选）？我不认为这个选项可以是可选的。事实上，我觉得在很多情况下这个比严重性还要重要。
- 顾客价值 (IM3XX)：这个事件如何影响顾客或者市场价值（可选）？再一次，我不认为该项只是可选的。
- 任务安全性 (IM4XX)：这个如何影响任务目标或者安全性（可选）？这个当然只是适用于特定的系统。
- 项目日程表 (IM5XX)：解决这个事件会如何影响项目日程表（强制性）？
- 项目成本 (IM6XX)：解决这个事件会如何影响项目成本（强制性）？
- 项目风险 (IM7XX)：伴随该事件修复的项目风险是什么（可选）？我认为这个也应当被分为强制。
- 项目质量/可靠性 (IM8XX)：伴随该事件修复的项目质量/可靠性影响是什么（可选）？这是另一个重要的变量。
- 社会的 (IM9XX)：伴随该事件修复的社会问题是什么（可选）？这可能只是和特定的系统有关。

以下列出了受影响的数据：

- 这个事件如何影响成本？这包括了分析的成本、如果修复完成之后的估计成本、如果修复未完成的估计成本，以及其他解决方案的成本。
- 这个事件如何影响时间？这包括如果修复成功需要的估计时间、如果修复成功的估计验证时间、如果修复没做完的估计时间，以及实际的实现时间。
- 这个事件的风险是什么？这应该是一个文本的描述。
- 这个对于日程表有什么影响？这包括假设事件被解决，假设没有被解决，以及如果被解决了，实际对日程表的影响是什么。
- 如果有可能，合同会做什么更改？

图 7-3 表示了一个使用分类信息来学习项目中有意思的事情的实例。这个巴累托图分析了 IVR 综合系统项目中的每个主要系统中的缺陷数量和百分比。这个项目叫做 NOP 项目，通过一个广域网、一个本地网、一个电话系统将 10 个系统连接起来实现一个大型分布式娱乐应用系统。

正如在图 7-3 中所见，交互式语音应答 (IVR) 应用大约占有一半的缺陷。客户端服务应用 (CSA) 大约占了 30% 以上。剩下的应用相对来说比较稳定。内容管理 (CM)

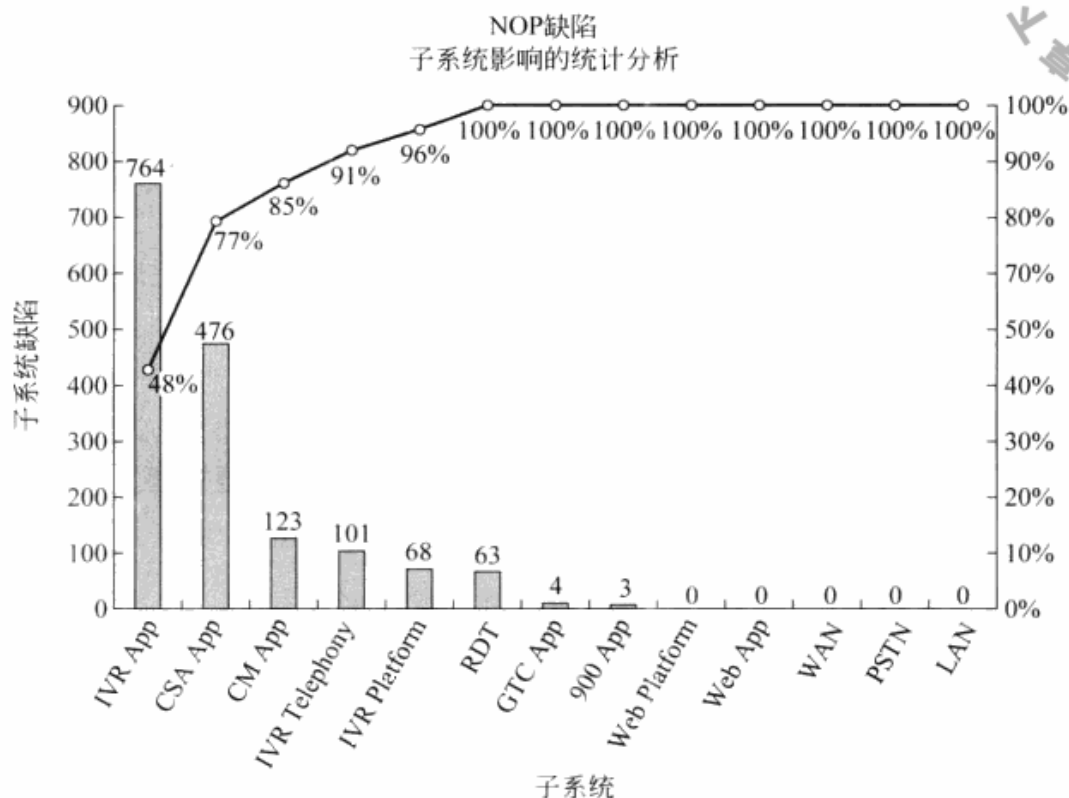


图 7-3 实例：分类

应用有超过 10% 的缺陷数量。交互式语音应答服务器的电话和 OS/硬件层每个大约占 5%，剩下的其他应用和基础占大约 4%。

这个数字告诉我们技术风险——发现缺陷的可能性——对于 IVR 应用来说是最高的。如果我们想要使用该信息来重新分配测试工作量，可以基于该信息更新我们的风险分析，然后检查更新的风险分析如何影响测试。如果我们想要使用其作为一个项目回顾，那么我们可能要进行深入分析来看这个应用中有多少缺陷。因此，我们在后续项目中可以做出哪些变更来减少 IVR 应用中的缺陷发生率呢？

### 7.4.1 缺陷域练习

假设将一个电话银行业务员团队参加 HELLOCARMS 测试作为一次 Beta 测试。银行业务员会输入客户实时申请，但是他们要在事后捕获信息并输入系统来保证 HELLOCARMS 系统没有缺陷影响到顾客。

银行业务员不是受过培训的测试人员而且也不愿意花费时间来学习测试基础。所以，为了避免银行业务员输入的书面事件报告质量太差以及在事件报告度量中引入争议，管理层决定，银行业务员发现问题后，将通过电子邮件通知测试分析师去输入报告。

以下是一个测试分析师收到一个银行业务员描述问题的邮件报告：

我在为一个有很好信用度的客户在房屋净值贷款应用中输入数据。她拥有一栋高价值的别墅，而且贷款额度还不是很大。

在屏幕上，HELLOCARMS 弹出“升级到由电话银行客户经理处理”的信息。但是，我单击了“继续”按钮后，在我没有输入任何的电话银行客户经理授权码的情况下，它



仍然允许我继续进行操作。

从这之后的客户应用中，所有的行为都是正常的。

我有另一个客户有一个相似的情况——高价值的别墅，中等大小的贷款额度——在那天晚些时候打进来电话。同样的问题，我不用输入任何授权码该系统就允许我进行处理。

回答以下 3 个问题：

- 银行业务员的电子邮件直接或者间接地提供了哪些分类和数据？
- 银行业务员的电子邮件直接或者间接地提供了哪些识别分类和数据？
- 什么信息丢失了，需要哪些后续调查？

这个问题在后面的缺陷生命周期报告中阐述。

如果是在一个教室中学习，那么你可以选择在一个 3~5 个人的小团队共同完成。

7.4.2 缺陷域联系报告

首先，评估每个恰当的识别和其影响分类和数据域以确定这个电子邮件或者其他我有的信息都已经列出来了。我的分析如表 7-2 所示。

表 7-2 事件报告 IEEE 1044 覆盖率

IEEE 信息	是 否 有
项目活动	假定我们知道所有此类 beta 测试的此项信息
项目阶段	很可能我们知道所有此类 beta 测试的此项信息
潜在原因	未知
可重复性	已知，但是需要对该问题做更多的分隔和复制
症状	已知
项目状态	未知，但是我们可以假设这个产品允许电话银行业务员越过类似风险管理政策是不可接受的
环境	很可能我们知道所有此类 beta 测试的此项信息
发起者	很可能在电子邮件的发送人信息中会有提供
时间	未知
承包商	某些承包商信息我们可以知道，但是其他的信息，像他们何时提供一次修复，是没有的
严重性	已知
优先级	未知，但是我们可以假设这里有一个高优先级
顾客价值	未知，但是至少可以推理出来
任务安全性	未知
项目日程表	在此时不合适，因为需要调查
项目成本	在此时不合适，因为需要调查
项目风险	在此时不合适，因为需要调查
项目质量/可靠性	在此时不合适，因为需要调查

IEEE 信息	是 否 有
社会的	在此时不合适, 因为需要调查
成本	在此时不合适, 因为需要调查
时间	在此时不合适, 因为需要调查
风险	不可用, 但是我们可以对该系统让电话银行业务员忽略银行风险管理政策做出一些推断以及描述该风险
日程表	在此时不合适, 因为需要调查
合同变更	在此时不合适, 因为需要调查

接着, 我在报告下面加了一些批注, 希望测试分析人员在我将报告放入系统前告诉我额外的信息。原始的信息是斜体字, 我的批注是正常字体。

*我在为一个有很好信用度的客户在房屋净值贷款应用中输入数据。*

我需要找出该顾客的确切数据, 包括: 收入、债务、资产、信用值等。

*她拥有一栋高价值的别墅, 而且贷款额度还不是很大。*

我需要找出这个别墅的确切价值和贷款数额。

我要测试不同的价值以及贷款额度组合来看我是否可以找到一个合适的方案。

*在屏幕上, HELLOCARMS 弹出“升级到由电话银行客户经理处理”的信息。但是, 我单击了“继续”按钮后, 在我没有输入任何的电话银行客户经理授权码的情况下, 它仍然允许我继续进行操作。*

我需要找出银行业务员是否在该域输入了所有项。

我将对空字符串、空格、有效字符但不是正确的验证码的情况进行测试, 同时执行其他的检查来看是否它完全地忽略了 this 域。

*从这之后的客户应用中, 所有的行为都是正常的。*

我会测试这个项目是否转移到贷款文件打印系统, 或者只是被忽略。如果它们被转移到了贷款文件打印系统, 贷款文件打印系统会知道这一步被忽略了以及进行处理吗?

*我有另一个客户有一个相似的情况——高价值的别墅, 中等大小的贷款额度——在那天晚些时候打进来电话。同样的问题, 我不用输入任何授权码就可以让我进行了处理了。*

这里我也想要找出这个申请人的确切信息、财产价值, 以及贷款数额。

## 7.5 度量元和事件管理

### 学习目标:

(K4) 分析多次创建的缺陷报告以及更新缺陷分类。

通常对测试分析人员来说, 在事件报告中使⽤文本性描述信息是很明显的, 但是对于分类的使用非常容易会混淆。我有一个客户告诉我他花费了上千美元用于咨询使用最新的分类主题——正交缺陷分类法, 来改进他们的事件管理系统, 但是, 他们试图在项目中省钱而不培训人们如何使用, 所以分类信息完全是没用的。



你应该从项目角度以及公司和过程改进的长期目标来分别看待事件分类信息。从项目角度来看，缺陷分类应当支持我们在基础大纲中以及本课程的早期章节讨论的测试过程监视。我们在一个项目中可以使用不同的度量元，像缺陷集群分析、缺陷密度分析，以及覆盖率（也叫做开启/闭合图）来管理缺陷趋势以及检查发布的可读性。

从公司和过程的角度上，我们想要评估做得怎么样以及如何可以做得更好。事件分类应当支持过程改进的新方案。我们应当可以评估阶段性的遗漏，也就是他们在同一阶段引入和消除的缺陷百分比。我们应当可以评估根本原因，这样我们才可以减少缺陷数量。而且，我们应当可以评估项目中的缺陷趋势来分析最好和最坏的实践在哪里。

图 7-4 是我提到多次的因特网相关项目关闭缺陷用时的一个实例。关闭用时图多次分析了缺陷趋势。图 7-4 表示了一些有意思的事情。在最初的 12 个星期——从 8 月 1 日到 10 月 17 日——平均缺陷关闭用时逐渐变高。在开始一个短暂的不稳定阶段之后，用时稳定在 11 天左右，但两个月之后变为 14 天左右。

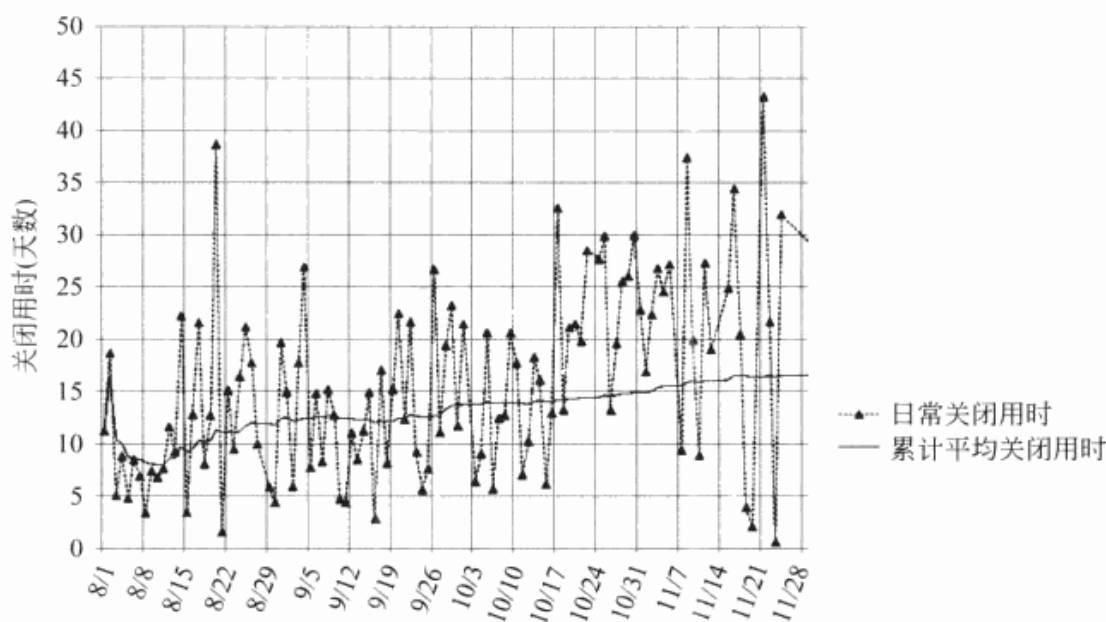


图 7-4 实例：缺陷关闭用时趋势图

但是，在 10 月 17 日，一定发生了某些中断。从 10 月 17 日到 11 月 17 日的一个月中，累计缺陷关闭平均用时又延长了 3 天到大约 17 天。之前累计缺陷关闭平均用时在项目两个月后延长了 3 天，但是现在它每个月都要再延长 3 天，增速是之前的两倍，记住，当项目继续的时候，累计缺陷关闭平均用时的延长或者缩短变得越来越困难，因为已关闭的缺陷报告数越来越大，因此，累计缺陷关闭平均用时是将历史数据的要点提了出来。

你可以看到在那个月里有超过 20 次日常关闭用时在累计平均关闭用时之上，而只有 4 次日常关闭用时在累计平均关闭用时之下。另外，超过累计平均关闭用时的日常关闭用时，很明显地高于累计平均关闭用时而低于累计平均关闭用时的那些日常关闭用时，只是稍低于累计平均关闭用时。因此，你不难想到日常关闭用时正在急剧加速累计平均关闭用时的延长。

发生了什么事？从趋势图来看没有确定的事情。趋势图表现了时间的趋势，而不是因果联系。为了在这种情况下找出答案，我着手运用两种调查的方式。首先，查看项目

的重要里程碑是否是在 10 月 17 日。这个可能会是一个新测试级别的入口，特别是一个可能会导致整个进程变慢的更加正式的测试级别。或者，是否引入了一个大的主干上的新功能，这可能会导致棘手的未被发现的问题。或者，开发团队的某些成员是否被重新部署到另一个项目。

第二种方法是在缺陷分类中。我会运行 3 个分析：

- 整个项目中受影响子系统的缺陷百分比分布。
- 从 8 月 1 日到 10 月 17 日这段时间内受影响子系统的缺陷百分比分布。
- 从 10 月 17 日到 11 月 17 日这段时间内受影响子系统的缺陷百分比分布。

在测试管理相关章节中，讨论测试过程监视和控制的度量元时，展示了这个图种类。

这个分析可能会揭示缺陷所在位置的变更。这样可能会导致下一轮的调查，最后，我们可能要调整缺陷分类、我们的风险分析，以及基于该信息调整我们对剩余测试的计划。

### 7.5.1 度量元和事件管理练习

这个练习遵守之前练习相同的标准。为了评审，假设将一个电话银行业务员团队参加 HELLOCARMS 项目的测试作为一次 Beta 测试。银行业务员会输入从客户那边得到的实时数据，但是他们也要捕获数据并将其输入现存系统之后来保证 HELLOCARMS 系统不存在影响顾客的缺陷。

银行业务员不是受过教育的测试人员而且也不愿意花费时间来学习测试基础。所以，为了避免银行业务员输入差劲的书面事件报告以及在事件报告度量中引入争议，管理层已经决定，当银行业务员发现一个问题，他们需要发一封电子邮件给测试分析师并由他输入报告。

一个测试分析师从一个银行业务员那里接收到以下描述问题的邮件：

我在为一个有很好信用度的客户申请房屋净值贷款而输入数据。她拥有一栋高价值的别墅，而且贷款额度还不是很大。

在屏幕上，HELLOCARMS 系统弹出“升级到由高级电话银行员处理”的信息。但是，当我单击“继续”按钮后，在我没有输入任何的高级电话银行授权码的情况下，它仍然允许我继续进行操作。

从这之后的客户应用中，所有的行为都是正常的。

另一个客户有一个相似的情况——高价值的别墅，中等大小的贷款额度——在那天晚些时候打来电话。同样的问题，我不用输入任何授权码就可以进行处理。

现在，对于这个练习，假设你看到了一些电子邮件和我这里展示的发送给测试分析人员的很相似，你会如何更新 IEEE 1044 分类法来获得信息并用于帮助分析这种问题的流程度？

### 7.5.2 度量元和事件管理练习参考答案

为了理解这个问题是多么的流行，使用帕累托分析法将这种失效类型和其他失效类



型进行比较是非常有帮助的。所以，你可以更新症状（RR5XX）的分类。这个分类需要对事件本身定义。通过加入这个特定的失效种类，即没必要升级到由电话银行客户经理批准的贷款，你可以接着做一些分析来比较该类失效和其他类型的失效，然后采取合适的行动。

## 7.6 沟 通 事 件

### 学习目标:

能回想起该部分内容即可。

低质量的事件报告是项目团队成员中关系变差以及产生冲突的主要原因。我的同事和我在测试团队评审的时候经常看到这种情况。为了维持团队成员间的良好关系，记住以下几点：

- 通常来说，抱怨或者强加指责不是测试人员的工作。避免任何有可能引起无端指责或者抱怨的陈述。
- 一个好的事件报告应当提供客观信息、坚持真相。如果你要做一个假设或者阐述一个理论，应该表明你这么做的理由。如果你确实决定做出这些假设或者理论，牢记第一个规则：不要针对特定的某人。
- 一个缺陷报告通常都是某件事情错误的断言。当你说存在一个问题，它会帮助进行改变，争取最大的精确性。
- 最后——这更像是一种思维倾向，但是真的很重要——开始着手把查看事件报告作为你能提供的服务，不只是提供给经理们还包括提供给开发人员。问一下开发人员，在你的报告中可以包含什么可以帮助他们的信息。由此造成的变化可能会让你感到吃惊。

某些测试人员在他们的缺陷没有修复的时候会感到很沮丧。当我们在一个评估中看到这种现象时，我的首要想法就是事件管理过程中有些事情搞砸了。理想情况下，一个缺陷修复或者事件修复会议包括一个跨职能团队的利益相关者来决定事件的优先级。如果只是按照开发人员或者测试人员的意见来决定什么应当被修复什么应当被推迟，肯定不好，那也不是说开发人员和测试人员的意见和投入付出，只是好的事件管理需要对事件处理的选项进行仔细的考虑。很少有项目奢侈到能够修复所有的事件。

总之，团队中良好的沟通和关系、良好的缺陷追踪工具，以及良好的缺陷修复对于一个好的事件管理过程来说都是非常重要的。事件管理是测试的一个基础，所有的测试分析人员和测试经理都要掌握它。

## 7.7 认证考试模拟题

在每个章节的最后，我们会准备一个或多个考试问题实例来加强你对该章内容的理解并用来准备将来的 ISTQB 高级测试经理考试。

1. 假设你是某项目中的一名测试经理, 该项目用来创建一个可编程恒温器用来控制家用的中央供热系统、空气流通设备和空调系统。除了正常的 HVAC 控制功能, 这个温度控制器也能够将数据下载并输入到在个人电脑中运行的一个基于浏览器的应用, 并将此用于后续的分析。

在质量风险分析中, 你发现基于浏览器的应用和不同的 PC 配置之间存在兼容性问题, 这非常有可能使该应用成为一个质量风险项。

你的测试团队现在正执行兼容性测试。考虑以下从兼容性缺陷报告的失败描述中获得的摘要:

- 1 将这个恒温器连接到一台 Windows Vista 个人电脑。
- 2 在个人电脑上启动该恒温器分析应用。应用正常启动后识别出该连接的恒温器。
- 3 试图从恒温器中下载数据。
- 4 数据没有下载。
- 5 试图 3 次下载数据, 数据不会下载下来。

基于以上的信息, 以下哪一个是缺陷报告中存在的问题?

- A 缺乏结构化的测试
- B 分类信息不充足
- C 分隔不充足
- D 重现缺陷的文档步骤比较差

2. 接上一题的场景, 你的测试团队还在执行一个兼容性测试。考虑以下从兼容性缺陷报告的失败描述中获得的摘要:

- 1 在一台 Windows XP 个人电脑上安装恒温器分析应用。
- 2 尝试启动恒温器分析应用。
- 3 恒温器分析应用没有启动。
- 4 进行 3 次恒温器分析应用的重新安装。恒温器分析应用在每次重新安装后都没有启动。
- 5 这个测试在之前的测试发布中通过了。

基于以上单独的信息, 以下哪个信息是该缺陷的最可能的假设?

- A 缺陷可能是一次衰退
- B 缺陷可能是时而发生的
- C 这个应用以前没有安装在 Windows XP 个人电脑上
- D 这个缺陷可能和其他缺陷重复



## 标准以及测试过程改进

美国铁路轨道宽度为 4 英尺 8 又 1/2 的标准源自于原始军标中对罗马帝王军用马车轮距的定义。军标……万岁！

——引自汤姆阿若，奥斯汀得克萨斯大学德语教授，  
见 [www.spikesys.com/Trains/st\\_gauge.html](http://www.spikesys.com/Trains/st_gauge.html)。

本章包括了两个部分，标准和测试过程改进。由于该章节第一部分标准和第二部分测试过程改进之间没有什么关系，因此整个章节的内容并不是很连贯。在标准这一部分中，大纲介绍了许多国际的、国内的和特定领域的标准。在测试过程改进这一部分中，大纲讲解了过程改进的基本概念，然后介绍了 4 种测试过程改进模型以及一种软件过程改进模型。本章包含 9 个小节：

1. 概述。
2. 需要考虑的标准。
3. 测试改进过程。
4. 改进测试过程。
5. 使用 TMM 模型改进测试过程。
6. 使用 TPI 模型改进测试过程。
7. 使用 CTP 模型改进测试过程。
8. 使用 STEP 模型改进测试过程。
9. 能力成熟度模型集成，CMMI。

现在，让我们一一讲解这些小节，看看它们与测试管理有什么关联。

### 8.1 概 述

#### 学习目标:

能回想起该部分内容即可。

无论是管理一个测试项目、一项测试任务，还是一个测试团队都是富有挑战性的。幸运的是，有各种好的主意和想法能够帮助您进行管理。

首先,存在许多标准适用于测试或影响着测试。标准有很多分类,如按行业区分、按安全关键程度区分以及按国家区分等。有些标准是法律强制要求遵守的或者按照市场上的现实状况必须遵守的。在某种程度上,您也许会将强制性的测试标准视为一种负担,不过至少你知道必须要做什么了。向我咨询的客户中,有些客户要遵守强制性标准,而有些则不用遵守,经过我的观察,那些要遵守强制性标准的测试团队比起其他不用遵守的测试团队更容易向管理层申请资金和获得其他方面的支持。

即使并不需要遵守强制性标准,您依旧可以参考这些标准,从中并获取一些有用的想法。不过,在采纳某一个想法之前,要弄明白标准中为何会这么规定,因为某些想法也许并不适合你的情况。

除了标准之外,还有许多技术能改进你的测试过程。和任何业务过程一样,测试过程也是可以改进的。即使是一个良好的测试过程,久而久之,也会变差。这并不一定是由于测试团队停止使用这个测试过程而造成的,也许是因为组织有了新的测试需求。

如果要着手进行测试过程改进,你可以使用一些通用的过程改进技术。另外,你还可以使用一些特定的测试评估和成熟度模型。有些人认为这些模型、标准和成熟度测量是繁重的、死板的、令人分心的、或者不适用的。的确,有些顾问、培训公司或其他组织将一些标准当万灵药卖,欺骗客户,但我们不能仅凭一些无耻的机会主义者的所作所为就认为这些标准没有用。你可以仔细地评估这些模型、标准和成熟度测量是否适合你的组织,然后取其精华,弃其糟粕。

事实上,那些拒绝使用所有模型、标准和成熟度测量的人是受到一种自我欺骗的影响,他们简单地认为所有这些都是“不适用的”,坚决主张他们的情况太特殊、非同一般,因此外来的想法在这里是行不通的。作为一名顾问,我到过世界各地数百个地方进行演讲,提供咨询和参与项目,我可以肯定地告诉您,各种人、项目以及产品的区别并不像某些人想象中的那么大。<sup>1</sup>

因此,在本章节中,我们将讲解一些标准、模型和成熟度测量,它们能帮助测试经理开展工作。

## 8.2 需要考虑的标准

### 学习目标:

(K2) 汇总相关标准的来源并解释在软件测试中的作用。

让我们先来快速地浏览一下各种主题的标准。这些主题直接或间接地与测试经理的工作存在关联。

- 软件开发生命周期。
- 软件测试与方法论。

---

<sup>1</sup> Kaner 等人在他们的“Lessons Learned in Software Testing”一书中用一种怀疑的眼光看待标准,但具有讽刺意味的是,该书的某些作者自身也是标准委员会的成员。



- 软件配置管理。
- 软件维护。
- 质量保证。
- 项目管理。
- 需求。
- 软件语言。
- 软件接口。
- 缺陷管理。

这些标准来自不同的地方。有些是国际标准，由国际标准委员会制定。有些是国家标准，由国家标准委员会制定，或者根据国际标准修改而成。另外，还有一些特定领域的标准，这些标准根据国际标准或国家标准修改而成，或者是专为该领域制定的。

对标准进行适当地选择和修改是很重要的。在考虑是否要使用一个标准时，确保您已经了解了这个标准的来源和历史，明白应该如何使用这个标准。对于标准，通常需要牢记以下几个方面。

我曾提到过，标准来自于各种不同国际的、国家的以及特定领域的标准组织。而归根结底，标准组织是由一群公认的专家组成的。不同于万有引力等物理定律，标准来源于这些专家的集体智慧、经验、实验证据，同时也包含了专家们的嗜好和偏见。因此，一个标准的可靠性和价值取决于其制定者。并不是所有强制性标准都是有价值的，所有的自律性标准都是没有价值的。

由于人们会不断地修改标准，生成新的版本。因此，当您在评估一个标准的时候，要注意所评估的是哪个版本。如果标准正在修订中，那么您至少应该看看修订信息，将其作为评估的一部分。当然，如果能获得标准的草稿那就更好了，您可以直接对这些草稿进行评审。

如何证实这些标准对您是否有用呢？如果要遵守的是一个强制性的标准，那么这个标准就是比较有用的，或许至少从以下 3 个重要方面来讲是这样的。首先，如果不遵守标准，那么您就无法正确完成工作，因此在某种程度上，标准起着参考手册的作用。第二点，回忆一下我们在第 3 章测试管理文档中进行的讨论，您也许会想起标准是测试方针和测试策略的有价值的参考文档。第三点，就是我刚才提到过的，这些标准能帮助您获得额外的测试支持以及资金投入。毕竟，如果强制性标准中规定了要进行一项特殊的测试活动，那么管理层对此还能有什么疑问呢？

即使是一个非强制性的标准，它依旧可以对您有所帮助。例如，标准可以帮助推广一种预防性的测试策略。另外，标准还提供了一些供参考的框架，如可调整的过程检验表、与测试技术相关的想法、潜在的测试目标等。可以将非强制性的标准作为构建测试结构的起点。

不过标准也可能妨碍测试工作。我提到过各种类型的测试策略，其中标准驱动的测试策略有一个弱点，即如果标准所适用的情况与您现在的情况相差很大，那么这个标准就是不恰当的。您最终会发现您没在做您应该做的事，而是在做一些其实您并不需要做的事。这两个错误的代价是您所无法承受的——第一种错误减少了有效性，而第二种错

误则降低了效率——如果您无法正确地使用或调整这些标准，那么就可能犯这样的错误。

如果同时要遵守数个标准，那么这个时候就有一个隐患。正如国与国之间、甚至城镇与城镇之间的法律是存在差异的，各种标准之间也是如此。有时候，标准之间仅仅是存在差异而已，但是在某些时候，标准之间就存在冲突。如果项目中有多个标准，那么要留心各种冲突和差异，如果发生这样的问题，可以对相应的标准进行调整。

重要的国际标准主要包括各种 ISO 标准和 IEEE 标准。以前 ISO 是指国际标准组织 (International Standards Organization)，不过，最近该组织将名称改为了国际标准化组织 (International Organization for Standardization)。与 ISTQB 组织的结构类似，该组织在各国都有分支结构。它推广以下与测试相关的标准：

- ISO 9126 标准，该标准描述了产品质量中与软件工程相关的各个方面。它包括 4 个子标准和技术报告。ISO 9126—1 标准描述了质量模型，包括我曾提到过的特性和子特性。ISO 9126—2 描述了外部的质量度量元，例如动态测试时可用的度量元。ISO 9126—3 描述了内部的质量度量，例如静态测试时可用的度量元。ISO 9126—4 描述了使用质量的度量元，例如用户使用时可用的各种度量元。
- ISO 12207 标准，该标准描述软件生命周期过程。基础级大纲中提到过该标准。
- ISO 15504 标准，该标准描述过程评估，有时候也被叫做 SPICE 模型。它与能力成熟度模型 (CMM) 以及能力成熟度模型集成 (CMMI) 有点类似。

IEEE 是指电气与电子工程师协会。这是一个美国的行业组织。不过，该组织有来自 100 多个国家的代表。它推广以下与测试相关的标准：

- IEEE 610 文档是 IEEE 组织使用的标准计算机词典。该文档包括了各种 IEEE 标准中使用的术语定义。注意，IEEE 610 中的术语与 ISTQB 的术语存在冲突。
- IEEE 829 软件测试文档标准。基础级大纲对这个标准进行了详细的讲解，而在本书中我们也再次提到了这个标准。
- IEEE 1028 软件评审标准。在基础级大纲中讲解过该标准。
- IEEE 1044 软件异常分类指导。在上一章节中我们曾提到过。

国家标准，正如你所想的那样，各个国家之间的标准是不同的，但这些标准依旧可以国际通用。比较典型的有英国 7925 标准。该标准描述了一系列测试设计技术，如下所述：

- 等价类划分技术。
- 边界值分析。
- 状态转换测试。
- 因果图测试技术。
- 语法测试。
- 语句测试。
- 分支/判定测试。
- 数据流测试。
- 分支条件测试。



- 分支条件组合测试。
- 改进的条件判定测试。
- 线性代码序列和跳转（LCSAJ）测试。
- 随机测试。
- 组件测试过程。

上述这些主题在我所著的高级软件测试分析师一书中详细的讲解。

其他标准还包括特定于业务领域的标准、特定于技术领域的标准、与测试和质量相关的标准等。例如，在航空电子设备这个行业中有一个 DO 178B 标准，该标准在欧洲也被称为 ED 12B 标准。其正式名称是“机载系统以及设备认证中的软件因素”。这个标准适用于民用飞机，不过我也曾看到有人在军用系统中使用。该标准描述的是软件相关的问题，包括飞机的嵌入式系统以及用于测试飞机嵌入式系统的软件。我们曾在测试管理这一章节中讲到过这个标准。

在航天工业中，欧洲航天标准化组织提出了 FAA 178B 标准，这个标准根据软件的安全关键程度向读者推荐各种方法和技术。标准中的技术包含：

- SFMECA——软件失效模式、影响和危急程度分析。
- SFTA——软件故障树分析。
- HSIA——硬件软件交互分析。
- SCCFA——软件常见故障原因分析。

医药系统相关的标准有美国食品及药物管理局标准，即美国联邦法规 21 章 820 款。该法规条款列出了一系列测试原则，包括：

- 期望的测试结果应该预先定义好。必须具备测试准则。
- 测试的目的不仅仅是通过覆盖恰当路径从而建立信心，同时还包括查找缺陷。因此，这个标准包括了以下几个方面的原则：一个良好的测试用例找到错误的概率较高，一次成功的测试应该能找到错误，仅仅检查那些常用的、典型的以及正确的用例是不够的。
- 为了避免作者偏见以及潜在的供应商问题，标准中要求测试人员应该独立于开发。
- 标准认为测试技能具有跨学科的特性，因此测试应包括应用以及软件这两方面的专家。
- 最后，该标准要求使用可重用的测试文档。同时还要求对测试状态尽可能进行审计以及独立确认，可以是周期性的或者由某些事件触发。

因此，上述原则能帮助我们定义自己的测试方针。不过，如果我们要遵守美国食品及药物管理局——美国联邦法规 21 章 820 款，那么在我们的测试计划中应该描述哪些具体任务？标准要求如下：

- 测试策划，包括第 3 章中的各个主题。
- 结构化的测试设计，包括语句、分支、条件/复合条件、循环、路径以及数据流。在我所著的测试技术分析师一书中将讲解这些技术。
- 功能测试设计，包括正常用例、所有输出、健壮性以及输出组合等。在我所著的

测试分析师一书中将讲解这些技术。

- 单元测试与详细设计之间的可追溯性、集成测试与概要设计之间的可追溯性、系统测试与需求之间的可追溯性。
- 一系列测试级别，包括单元测试、集成测试、功能测试、系统测试，以及验收测试。
- 缺陷修复分类小组应该包含正确的利益相关者。
- 结果评估与最终报告。

除了上述标准之外，还有其他标准可用，确切地说，还有很多其他标准。有些可能是特定于公司的。有时候，一个公司或小组也许会将一个宽泛的行业标准修改之后拿来使用。作为一名测试经理，无论团队正在使用什么标准，必须确保你们正在按照标准正确地工作。

## 8.3 测试改进过程

### 学习目标：

能回想起该部分内容即可。

ISTQB 基础级大纲以及高级大纲的主要的主题之一便是测试能改进软件。这主要从两方面进行。一方面，测试人员在项目早期的参与能预防缺陷。在另一方面，在测试执行过程中测试能发现一些重要的缺陷。然而测试并不是改进软件的唯一途径。通过使用更完善的软件过程也能做到这一点。由于测试过程自身也是软件过程，因此测试过程是可以改进的。这些改进能使测试更有效、更高效。

既然您正在阅读一本测试管理书籍，那么也许听说过例如 CMMI 等软件过程改进模型。如果您和我一样，也对这些模型做过研究，那么应该知道这些模型中与测试相关的部分并不多。这些模型中的不足促使很多人专门设计了测试过程改进的模型，这其中也包括我。

在接下去的 4 个小节中，我们将讲解 4 个主要的测试过程改进模型：

- 关键测试过程（Critical Testing Processes, CTP）。
- 系统化的测试与评估过程（Systematic Test and Evaluation Process, STEP）。
- 测试成熟度模型（Test Maturity Model, TMM）。
- 测试过程改进（Test Process Improvement, TPI）。

我对上述模型按照字母而非使用频率进行了排序。与这些模型相关的 4 本书籍还在销售中，买的人也挺多的，这从一方面也说明了目前在很多组织中正在应用这些模型。<sup>2</sup>

下面还有 4 个测试过程模型，不过在此我并不会详细讲解，如果对它们感兴趣，您

---

2 我的“Critical Testing Process”一书描述了关键测试过程模型。Rick Craig 的“Systematic Software Testing”一书描述了系统化的测试与评估过程模型。Ilene Burnstein 等人的“Practical Software Testing”一书描述了测试成熟度模型。Martin Pol 和 Tim Koomen 的“Test Process Improvement”一书描述了测试过程改进模型。



可以自行研究。

- 测试组织成熟度 (Test Organization Maturity, TOM)。
- 测试改进模型 (Test Improvement Model, TIM)。
- 软件质量等级 (Software Quality Rank, SQR)。
- TMap。

我说过, 整个软件过程都可以并且也应该进行过程改进, 包括测试。过程改进的基本思想是从我们犯过的错误中吸取教训。每次我们做一件事情的时候, 如果我们有所留意, 那么总能发现一些可以改进的地方。

质量控制专家休哈特曾提出过一个过程改进的过程, 如图 8-1 所示。后来戴明对这个方法进行了推广, 因此人们将这个方法称为戴明循环或休哈特循环。<sup>3</sup>

这个循环的基本思想是过程改进包含了 4 个线性并且迭代的步骤:

- 计划, 思考如何改进过程。
- 执行, 改进过程。
- 检查, 评估过程改进的效果。
- 操作, 根据评估结果采取相应的措施, 例如进行下一次迭代的计划。



图 8-1 戴明或休哈特循环

过程改进的根本思想是生产产品的过程自身的质量对产品质量有很大影响。我们可以通过使用更好的过程来生产更加完美的软件, 而不是仅仅到过程的最后阶段测试执行时去查找缺陷、消除缺陷, 或者更恶劣的是, 完全将这些缺陷丢给用户或客户。我们在第 3 章曾讲过, 过程改进能提高产品的质量, 从而提供商业上的回报, 在项目中越早进行过程改进, 那么就能以越低的成本获取相同的回报。

戴明循环为我们提供了一个用于过程评估和改进的框架。我们可以从任何已有的过程开始, 然后对其中的一个或多个过程使用计划—执行—检查—操作方法, 久而久之, 这些过程就会变得更加完善。

在大多数情况下, 这种方法提供的是增量式的改进。另外, 只能改进您所拥有的过程, 因此如果不是在做应该做的事情, 这种方法是无法察觉到这个问题的。还有, 虽然您知道戴明循环能让您做得更好, 但是您怎么去实时地判定当前是做得更好了还是更差了呢?

过程模型能提供一个起点、一个标准框架以及一种测量过程的方法。通过使用过程模型, 我们可以对过程进行评估, 然后找出当前过程中需要改进的地方。

在某些情况中, 如果要使用过程模型进行评估, 那么先要对您现有的过程的成熟度进行评级。由于成熟度评级模型是一维的, 或者在有些时候, 是以一个过程接着一个过程的形式表现的, 所以通常我将这些模型称为规范性模型。之所以这么称呼它们, 是因为这些模型规定了对各种过程进行改进的顺序。大纲将这些模型称为过程参考模型。

3 要详细了解戴明在质量管理方面的想法, 可以参考 Mary Walton 的 “The Deming Management Method” 一书。

### ISTQB 术语

**测试成熟度模型 (TMM):** 测试过程改进的五级阶段框架, 它与能力成熟度模型 (CMM) 相关, 后者描述了有效测试过程的关键要素。

**测试成熟度模型集成 (TMMi):** 与能力成熟度模型集成 (CMMI) 相关的五层测试过程改进框架, 描述了有效测试过程的关键因素。

**测试过程改进 (TPI):** 用于测试过程改进的一个连续框架, 描述了有效测试过程的关键要素, 特别针对于系统测试和验收测试。

我对规范性模型存在疑问, 作为一个咨询顾问, 我认为业务的改进应该是基于这些改进所具有的商业价值或者是基于这些改进能缓解的组织缺陷。一个一维的成熟度评级模型可能导致业务改进只局限于整个软件过程的一小部分, 或者由于模型规定的顺序, 所以过程中重要的部分未能得到及时的改进。

我更喜欢非规范性的过程模型。这些过程模型描述了重要的软件过程, 以及在这些过程中要做些什么, 但是它们并未规定应该以什么样的顺序进行过程改进。

不过这并不意味着使用非规范性过程模型进行评估就无法给出建议以及指导人们应该按什么顺序实现这些建议。当我和我的助手为客户进行评估的时候, 我们的报告中就明确地包含这两者。关键是, 你可能会发现我们的任意两份报告给出的建议可能是非常类似的, 不过实现的顺序可能就大不相同了。为什么呢? 因为每个客户的具体情况是不同的。有时候, 限制或前置条件可能会影响这些建议的实现顺序。每个组织存在的限制以及它们的具体情况都是唯一的, 而一个非规范性的模型能适应并满足这些具有唯一性的组织需求。

顺便提一下, 大纲中提到的非规范性模型有内容模型等。

无论你使用哪种方法, 不要将过程评估作为一种一次性的活动。在经过评估之后找到了可改进的地方, 你就可以开始着手进行改进。然后在未来的某个时候, 你应该再次进行评估, 检查改进的效果并且对你的过程改进进行校正, 这个过程就是在已有的过程模型上应用戴明循环的过程。

## 8.4 改进测试过程

### 学习目标:

(K3) 与相关人员一起按照通用步骤编制测试改进计划。

可以将测试过程改进模型作为提升测试成熟度以及专业化过程的一部分。通过采用标准的模型, 组织能获得可比较的度量元以及测量。也就是说, 他们能在公司内部比较各个项目, 比较公司内不同小组的项目以及与使用相同模型的其他公司的测试小组进行比较。

我曾提到过, 模型主要有两种类型。第一种是阶段式的或规范性模型。这些模型要



求对过程的成熟度进行测量或可能要收集每个主要过程区域的成熟度数值。虽然我怀疑这些成熟度数值对顺序的过程改进工作是否有意义，不过它们使得人们能比较各个公司和组织的过程成熟度，当然这有一个前提条件，即评估人员能一致地测量各种组织的成熟度数值。

第二种是持续的或非规范性的模型，这类模型不会给出改进的顺序。它们允许组织根据他们自身的业务需求将注意力集中在高优先级的问题上。

在本书中将讲解 4 个测试过程改进标准模型：

- 关键测试过程 (CTP)。
- 系统化的测试与评估过程 (STEP)。
- 测试成熟度模型 (TMM)。
- 测试过程改进 (TPI)。

前两个，关键测试过程 (CTP) 与系统化的测试与评估过程 (STEP) 是非规范性的。后面两个，测试成熟度模型 (TMM) 与测试过程改进 (TPI) 是规范性的。

所有这 4 个模型都能用于对测试进行评估。不过，在评估结束后，TMM 和 TPI 会给出过程改进的一个规范性顺序。按照这个顺序进行改进能让你的测试过程稳定地、增量地提升其成熟度。有趣的是，这两个成熟度模型建议的改进顺序是不同的。相反地，STEP 和 CTP 模型能让组织自行确定改进的顺序，以获得最高的投资回报。

在某种程度上，规范性的模型更易于使用。由于规范性模型的评估人员无需对各个过程域中的可改进之处进行排序，因此这种模型的评估过程通常而言会简单很多。而对于非规范性模型而言，其评估人员必须设计一个业务用例，并且这个用例的准确度和精确度都要达到一定的程度，以便确定可改进之处的优先顺序以及过程改进的整体路线图。

然而，我认为（这仅仅是我的观念，与 ISTQB 高级大纲无关）非规范性模型考虑到了业务的成本和收益，这使得组织更乐意对过程改进给予支持。作为一名咨询人员，我和众多客户一起进行过程改进，一个有良好收益预期的计划比一个仅仅认定组织必须以一定的顺序进行过程改进的计划更可能获得通过并得到资金支持。

### 8.4.1 一个通用的过程改进框架

现在让我们来看一个通用的过程改进框架，这个改进框架进一步细化了计划—执行—检查—操作这些戴明循环中的步骤。

- 初始化：在这个步骤中定义了过程改进的目标、范围，以及覆盖。如果计划使用标准的过程改进模型，那么在这个时候您应该做出选择。如果想使用一个自定义的模型，那么在这个阶段您就应该开发模型，或者改进模型。如果您所选择的模型未提供用于评估结果的度量，那么您应该根据目标，自定义一些度量元用于评估过程改进是否成功。与测试方针相似，您的过程改进也应该有一个方针，并且利益相关者都同意并支持这个方针。
- 测量：进行评估。如果这是第一次评估，那么有时候也被称为建立基线。通过与基线进行比较，可以评估任何变更的效果。同时，通过这个评估，你又可以发现一系列可改进的地方。

- 排序和计划：在这个步骤中，将定义过程改进的顺序。如果是规范性的模型，那么这个顺序已经定义好了。而对于非规范性模型，应该考虑以下因素并对过程域进行排序，例如，投资回报；风险和限制；与组织策略、目标以及项目的合规性；定量的或定性的收益；目前的过程所造成的组织上的病痛。<sup>4</sup>在定义好过程域的改进顺序后，就可以指定并启动改进计划。
- 定义和重定义：执行过程改进。包括定义新的过程和更新现有的过程，并准备好部署经过改进的过程。
- 执行：部署改进后的过程。这个步骤需要考虑的因素包括：组织以及相关人员的准备程度、限制、政治因素等。常见的活动包括培训和指导相关人员、使用改进后的过程执行一些试点项目，以及在组织内全面推广这个过程。
- 确认：在部署了改进后的过程之后，可以使用度量元对这些改进进行评估。在这个时候，您可能会发现，过程改进的结果比期望中的要好，或者与期望中的一样好，或者没有期望中的那么好，或者更坏的情况，过程改进完全失败了。如果改进的结果并没有想象中的那么成功，那么显然您还要花更多的精力去进行过程改进。
- 优化：在这里您要决定接下去的步骤。您可能决定对当前的过程重新执行这些改进活动，特别是当没有获得预期结果的时候。如果您使用的规范性模型内包含了一个成熟度模型，那么可以使用这个模型评估你们当前的成熟度。然后，可以决定是否要改进下一个过程。

可以将这个模型与标准的模型混合使用，或者也可以单独使用这个通用模型。这个模型能让您专注于特别重要的测试过程。当然，也可以直接在当前的测试过程中应用这个改进模型，通过对比当前过程的有效性和效率以及项目完成后的有效性和效率，就能得出实际改进的质量和结果。

### 8.4.2 案例研究：测试评估的结果

我在 RBCS 的同事以及我本人为很多客户做过评估。我们通常会使用关键测试过程框架，稍后我会解释这个框架。图 8-2 是 RBCS 评估报告的摘要，我和 RBCS 印度分支的员工一起为一位客户进行了评估，这位客户在美国和印度都有分支机构。在这份报告的一开始，我们先列出了一系列按重要性排序的建议。图中给出了一些高优先级的建议。建议您在继续阅读本节之前先仔细阅读一下图 8-2。

从图中可以看到，这位客户过度依赖于系统测试来检查并移除缺陷。因此，在这个阶段测试团队发现了很多缺陷。然后，开发团队修改了大部分的缺陷。我们在第 3 章讲过，在这个阶段发现并解决缺陷比软件投入使用后再解决要节约很多成本，但是，与在生命周期的早期发现并解决缺陷相比，花费成本就会大很多。在系统测试阶段，开发部门无法处理所有的缺陷会导致发布延期以及产生大量的加班费用。

---

<sup>4</sup> 组织上的病痛是需要考虑的，因为您必须将这个改进计划推销给组织。销售的一个基本原则便是，比起获得一些改善，人们愿意花更多的精力，同时也更迫切地逃离痛苦。



问题：过度依赖系统测试检查并移除缺陷浪费金钱，导致发布推迟，恶化项目结束阶段的关键决策。

建议：

确保每个开发任务有一个相应的代码评审和单元测试，因为代码评审和单元测试可以防止大量缺陷进入系统测试阶段。

继续规范需求和设计过程，包括合乎规范的评审。

专注于通过需求评审预防缺陷以及早期单元测试阶段的消除缺陷。

收集每个主要项目阶段引入缺陷和消除缺陷的百分比度量元，采取行动尽可能从一个阶段逸入下一阶段的缺陷百分数降低为 0。

对所有项目采用统一的过程和阶段规范缺陷跟踪。

图 8-2 改进建议

对于这个问题，我给出了 5 个建议。

1. 确保每一项开发任务都包含了代码评审任务和组件测试的任务。不过数据表明，除非您的组织使用行业中最优秀的实践方法，否则代码评审和组件测试并不能达到您期望中的效果，有很多缺陷会被遗漏，从而进入系统测试阶段。

2. 将需求和设计过程正规化。制定需求规格说明的过程是缺陷的主要来源。需求规格说明以及设计规格说明中的漏洞，尤其是非功能需求中的漏洞，会导致大量缺陷。因此，该组织需要一个更优秀的过程，包括引入恰当的评审，以减少引入的总的缺陷数目。

3. 强调需求评审阶段要预防缺陷或组件测试阶段要发现缺陷。这句话的意思是，仅仅执行了需求评审和组件测试是不够的，我们希望在这些活动中预防缺陷、发现缺陷。

4. 收集每个主要的项目阶段的缺陷数据，包括引入的缺陷百分比和解决的缺陷百分比，然后想办法将每个阶段遗漏的缺陷尽可能降到零。这里我建议使用一个度量的缺陷管理过程，这样就能知道哪里引入了质量问题，哪里遗漏了大量缺陷。

5. 在所有项目中采用一个统一的缺陷跟踪过程，并且在同一个阶段引入正式的缺陷跟踪。在第 7 章中我们讲解过一些正式化的缺陷跟踪过程，在这些过程中我们并不需要收集所有的缺陷度量元。早期的缺陷检测活动，例如需求评审、设计评审、组件测试，可以使用一些非正式的技术跟踪缺陷和收集度量元，而在正式的测试阶段，例如系统测试和验收测试阶段就可能需要使用更多的正式技术。

除了这些建议之外，我还提供了一些额外的支持，如业务用例等。

## 8.5 用 TMM 改进测试过程

### 学习目标：

(K2) 总结 TMM 中定义的测试改进过程。

(K2) 论述测试改进模型 TMM 中的评估准则。

测试成熟度模型，或 TMM，是由 Ilene Burnstein 和她的同事们一起发明的。Burnstein 博士是伊利诺伊技术学院的教授。测试成熟度模型于 1996 年首次发表。

Burnstein 与她的同事开发 TMM 模型是为了能与 CMM 模型互补与一致。这里所说的“互补”是指补充 CMM 中与测试相关的内容。“一致”是指 TMM 保留了 CMM 中使用的 5 个成熟度级别，每一个测试成熟度级别与 CMM 中的开发成熟度级别是类似的。

CMMI 模型正在逐渐替代 CMM 模型并受到广泛应用，同样的，TMMi 也正在替代 TMM。TMMi 模型是基于 TMM 模型的，并作为 CMMI 模型的一个补充。它使用了 CMMI 中的结构来组织以下内容，包括过程域、通用目标、通用实践、特定目标和特定实践。TMM 将测试定义为所有软件质量相关的活动。例如，需求评审被认为是一种测试活动。这与 ISTQB 基础级和高级的大纲是类似的。

我曾提到过，TMM 有 5 个成熟度等级，用于评估和改进。每一个级别包含了一系列定义好的过程域。在进入下一个级别以前，组织必须完全满足每一个过程域中的关键实践。这种方法有时候也被称为阶段式表示法。按照 ISTQB 高级大纲的说法，TMM 提供了一个过程参考模型，同时也提供了一个内容参考模型。也就是说，如果您想要以阶段式的方式进行改进以达到期望的成熟度级别，那么可以将 TMM 当作一个规范性的模型使用，而如果你想要解决某些特定的业务问题或要识别并解决组织中存在的问题，那么也可以将 TMM 作为一种非规范性模型使用。

现在，让我们来看看 TMM 的 5 个级别，以及与其相关的特性和过程域。第一个级别被称为初始级。在这个级别中，测试是混乱的、未定义的。很少撰写测试相关的文档、没有真正的测试过程，测试没有什么结构以及没有测试级别的概念。测试活动在编码完毕之后立刻进行，通常是比较随意的，不使用任何测试设计技术。无论从组织上来看还是从过程上来看，测试都只是调试的一部分。通常测试没有什么明确的目标，但在某种程度上，人们假定测试能证明软件可以正常工作。由于没有测试过程，因此也就没有什么关键过程域能“达到”这个级别。

注意，如果使用的是应对性的测试策略，管理又很糟糕的话，可能就是处于这个级别。另外，敏捷开发方法，如果没有良好的测试实践与其配合，也会陷入混乱的沼泽。

第二级被称为定义级。在这个级别，已经定义了什么是测试，并从过程上以及组织上将其与调试明确区分开。为了达到第二级，组织必须定义各种过程，用于建立测试方针和目标；设计并执行测试计划；应用正式的测试设计、实现和执行技术以及方法；建立和维护一个恰当的测试环境。

第三级被称为集成级。在这个级别中，测试已经完全被集成到软件生命周期中，这与本书以及 ISTQB 基本的测试过程所描述的是一致的。测试过程已经文档化，并遵守恰当的标准。为了达到第三级，组织必须定义各种过程，用于确保为员工提供正确的测试培训、建立一个测试过程并将其集成到软件生命周期中，以及在测试执行期间进行测试监控。同时还应具备一个独立的测试组织。

第四级被称为管理级和测量级。在这个级别中，已经完全定义了测试，符合项目、产品以及组织的需要，并且是可测量的。测试包括一些预防性的活动，比如评审或审查。为了达到第四级，组织必须定义各种过程，用于确保执行了同行评审、正确测量测试结果，以及使用这些测量数据进行有意义的软件质量评估。

第五级被称为优化级。在这个级别，已完全定义测试，并且能用恰当的度量元评估



其有效性和效率。通过使用这些度量元，还能对测试进行不断优化。为了达到第五级，组织必须定义各种过程，用于收集和使用有效数据，为缺陷预防、测试过程优化以及持续的质量控制提供支持。

注意，在这个模型中其实暗含了一个哲学理念，即原则优先。应该在独立的测试团队建立之前先定义测试方针。在 RBCS 为客户进行测试评估的时候，我们很少看到组织的测试成熟度是以这样的方式发展的。不过，如果你的组织希望其成熟度能以一种“从上至下”的方式发展，那么这个模型就比较适合。

还有，这个模型强调定义以及文档化。因此这个模型可能导致测试团队将精力放在生成大量与过程和工作产品相关的文档上。在某些受管制的行业中，可能比较适合使用这个模型。

如何使用 TMM 进行改进以及评估？您可以对一些关键过程域进行评估。对于每个关键域，需要达到一系列预定义的成熟度目标和子目标。根据是否满足了特定过程域，就可以对当前的成熟度进行评估。

目标和子目标被定义为大概的活动、特定任务以及职责。通过与经理、开发人员、测试人员、客户和用户交流，可以评估是否已经达到了这些目标和子目标。

记住成熟度模型是阶段式的。阶段式模型有两个含义。第一，除非已经达到了一个过程域的所有目标和子目标，否则这个过程域是未被满足的。因此，一个过程域要么是未被满足的，这意味着我们无法确定是否已经实现了任何改进，要么是已被满足的，这意味着我们不必再对它进行改进。

由于过程域只有满足和不满足这两种状态，因此这极大地简化了评估的过程。不同的评估人员评估同一个组织所获得的成熟度数值应该是一样的。然而，这种将复杂的问题简化到非“是”即“否”方法，可能导致遗漏一些重要的特征。

阶段式模型的第二个含义便是为了到达更高的级别，必须满足当前级别定义的所有过程域。例如，假设除了未定义一个测试方针之外，已经满足了第二级和第三级的所有过程域。在这种情况下，你的成熟度还是停留在第一级，即初始级，因为你还没有满足第二级定义的所有过程域，更不要说第三级集成级了。

类似的，如果除了还没有定义一个培训项目之外，你已经满足了第二、三、四级中所有其他过程域。在这种情况下，成熟度还是停留在第二级定义级，因为没有满足第三级集成级的所有过程域，更不用说第四级管理和测量级了。

这就是为什么我将这个模型称为规范性模型的原因。为了连续地、平滑地、增量地提高成熟度，达到最高级，即第五级优化级，必须按照 Burnstein 和她的同事规定的顺序改进过程。如果你期望的模型是自上而下的、重视文档化的，那么这种顺序的改进适合你。换句话说，如果觉得 Burnstein 描述的这个哲学理念和方法很适合您的组织，那么 TMM 就是适合您的。

在很多情况中，TMM 的哲学理念以及顺序的过程改进模型是有意义的。例如，在受监管的行业中按照合同规定的要求进行开发，在这种情况下，要提前明确测试的目标，

并建立一个稳定的、可审查的文档记录。<sup>5</sup>

## 8.6 用 TPI 改进测试过程

### 学习目标:

- (K2) 总结 TPI 中定义的测试改进过程。
- (K2) 论述测试改进模型 TPI 中的评估准则。

测试过程改进模型或 TPI, 是由 Tim Koomen 和 Martin Pol 根据他们为不同客户服务的经验总结而来的。Koomen 和 Pol 曾参与了双子星公司的 TMap 模型的开发。虽然在本书中或高级大纲中并没有直接讲解 TMap 这个模型, 不过 TPI 模型本身就是基于该模型的。

与 TMM 相比, TPI 是一个更加细化的模型。一方面, 这个模型将测试过程分为更多的过程域。另一方面, 它为大多数的过程域定义了多级别的成熟度, 而不是只有满足和不满足之分。我稍后会讲解这一点。

与 TMM 一样, 可以使用 TPI 模型测量测试过程的成熟度并给出改进建议。TPI 也是一个规范性模型。

不过, 与 TMM 不一样的是, TPI 并不与 CMM 模型匹配。同时, 它也不匹配 ISO 15504 标准。当然, 既然 TPI 与 CMM 不匹配, 那它与 TMM 也就不匹配了。

TPI 有 20 个关键过程域。其中有 18 个过程域与下列任意一项相关, 下列 4 项被称为测试过程的基石:

- 生命周期 (Lifecycle)。
- 组织 (Organization)。
- 基础设施和工具 (Infrastructure and tools)。
- 技术 (Techniques)。

另外两个过程域和上述所有 4 个基石都相关。测试过程基石的概念来源于 TMap 模型。

TPI 中的每个过程域分为 1 个、2 个、3 个或 4 个成熟度级别。同时每个过程域都有一个级别为 0 的成熟度级别。对于那些有 4 个成熟度级别的过程域, 其级别可以从 A 级 (第一级) 一直上升到 D 级。

根据 TPI 成熟度矩阵的定义, 可以使用过程域的成熟度级别来确定整体的成熟度级别, 从 0 级到 13 级, 如图 8-3 所示。这个图也列出了每个过程域划分的级别数目。我们可以将这些成熟度级别划分为 4 个级别, 从初始级到优化级。

TPI 识别了 20 个过程域, 可以按测试过程的 4 个基石来对这些过程域进行分类。对于生命周期这个基石, 有 3 个测试过程域:

---

<sup>5</sup> 与 TMM 相关的信息, 可以参考 Ilene Burnstein 等人的 “Practical Software Testing” 一书。与 TMMi 相关的信息, 可以浏览 [www.tmmifoundation.org](http://www.tmmifoundation.org)。



分类	关键域	0	1	2	3	4	5	6	7	8	9	10	11	12	13
L	测试策略		A					B				C		D	
L	生命周期		A			B									
L	介入时间			A				B				C		D	
T	估算和计划				A						B				
T	测试规范技术		A		B										
T	静态测试技术					A		B							
T	度量元					A	A		B				C		D
I	测试自动化				A				B			C			
I	测试环境				A				B						C
I	办公室环境		A												
O	承诺与激励		A				B						C		
O	测试职能和培训				A			B			C				
O	方法					A						B			C
O	沟通			A		B							C		
O	报告		A			B		C					D		
O	缺陷管理		A				B		C						
O	测试件管理			A			B				C				D
O	测试过程管理		A		B								C		
A	评估							A			B				
A	低级别测试					A		B		C					

图 8-3 TPI 测试成熟度矩阵

- 测试策略。
- 生命周期。
- 测试介入的时间。
- 估计和计划。
- 测试规格说明。
- 静态测试。
- 度量元。

对于基础设施和工具这个基石，有 3 个测试过程域：

- 测试自动化。
- 测试环境。
- 办公环境。

对于组织这个基石，有 8 个测试过程域：

- 承诺和激励。

- 测试职能和培训。
- 方法论。
- 沟通。
- 报告。
- 缺陷管理。
- 测试件管理。
- 测试过程管理。

有两个测试过程域涵盖了所有的基石：

- 评估（验证和确认）。
- 低级别测试（单元测试和集成测试）。

在图 8-3 中，20 个过程域被排列在同一个矩阵中，在这个矩阵中，它们各自的成熟度级别和整体的成熟度级别是联系在一起的。我会一一列地解释这个矩阵。最左边的这一列是一个代码，帮助你记忆这个过程域对应哪个基石。第二列是测试过程的名称。这个图的主体有 14 列，在第一行标题行中，分别是 0~13 的这些列。图中的每一行是每个过程的一系列字母。每一行至少包含一个 A。办公环境只有一个 A，在标题为 1 的列中。这意味着，办公环境这个过程域，与 TMM 的过程域类似，只有满足或不满足这两种情况。

再来看看度量元和测试管理这两行。每一行包括 4 个字母，从 A 到 D。也就是说，有 4 个级别的满足程度。我们最好能够逐渐地、并行地改进每一个过程域。

因此，这个表中共有两个维度的成熟度：

- 各个过程：每个过程都有某种成熟度等级，共有 4 种不同的等级划分：无或 A；无、A 或 B；无、A、B 或 C；无、A、B、C 或 D。
- 整体：根据每个过程的成熟度，我们可以评估整体的成熟度。我们可以根据第一行中的 0~13 这些数字从小到大一列一列地进行检查，直到发现有过程域未达到规定的成熟度级别。

让我们来看看具体是怎么做的。如果你的整体成熟度从级别 0 开始。要提升到级别 1，需要做到以下改进：

- 将测试策略过程域提升为 A。
- 将生命周期模型过程域提升为 A。
- 将测试规格说明技术过程域提升为 A。
- 将办公环境过程域提升为 A。
- 将承诺和激励过程域提升为 A。
- 将报告过程域提升为 A。
- 将缺陷管理过程域提升为 A。
- 将测试过程管理过程域提升为 A。

现在为了要从 1 提升到 2，需要做到以下改进：

- 将介入时间过程域提升为 A。
- 将沟通过程域提升为 A。



#### ■ 将测试件管理过程域提升为 A。

可以看到这些列中的颜色从左至右为深灰色、白色、浅灰色，以及深灰色。这些颜色的变化表示 TPI 将整体的成熟度又划分为 4 个等级。

第一个整体成熟度等级 0，被称为初始级。这个级别和 TMM 中的初始级是同样的含义，即非正式的、混乱的。TPI 和 TMM 中的初始级这个成熟度等级描述的没有结构和过程的情形实际上是指测试过程没有满足模型作者的要求，即某些过程域要达到一定的成熟度。

这个模型对我而言是有参考价值的。不过让人困惑的是，Koomen 和 Pol、Burnstein 和她的同事们对于改进哪些过程域以及改进到什么程度，以逃离初始级这个测试过程的地狱的看法是不一致的。

接下去的一个整体成熟度级别 1~5 级，被称为可控级。在这个级别中，测试过程被分为不同的测试级别，并已制定了测试策略和计划。使用了充分的测试规格说明技术。您记录、报告并管理缺陷。您的测试件和测试环境都是受控的。您为测试人员提供培训。

再下面一个整体成熟度级别为 6~10 级，被称为高效级。在测试过程可控以后，就要提高效率。根据 TPI 的定义，有一系列方法可以提升效率。您可以将测试自动化。您可以将测试过程更好地集成到各个测试级别，同时还可以将测试更好地集成到软件生命周期中。

最终的一个整体级别 11~13 级，被称为优化级。在这个级别中，可以不断地进行测试过程改进。要使用数据来驱动这个改进的过程。

与 TMM 类似，TPI 中也暗含着一个哲学理念。TPI 强调先建立一个基础的过程，然后再逐步进行矫正、改进、提高效率。例如，一个使用 TPI 模型的测试团队可能在高效级才会定义一个测试方针，而在 TMM 模型中这个活动要早得多。因此 TPI 是一个“自底向上”的模型。使用这种理念来提升测试成熟度的组织更常见一些，虽然通常他们都是无意识地在这么做，而非精心设计的。

注意，与 TMM 模型不同的是，TPI 模型并不强调定义和文档化。因此使用这个模型的测试团队能在一个轻度文档化的环境中更高效地进行测试。

那么我们如何使用 TPI 模型进行评估和改进呢？若要进行评估，我们要根据 TPI 模型中定义的成熟度检查点对所有 20 个关键过程域进行检查。TPI 的评估包括定量度量和定性的会谈。

如果对于一个特定的成熟度级别，所有的检查点都已经满足，那么这个过程域就已经达到了该成熟度级别。假设报告这个关键过程域的所有级别 A 和级别 B 的检查点都已经满足了，那么报告过程域达到了成熟度 B 级。

我曾说过，TPI 的持续改进的颗粒度更细。高级大纲将这称为连续式表述，与 TMM 模型的阶段式表述相对应。TPI 的连续性体现在每一个过程域有不同的成熟度级别。

然而，TPI 是规范性的。整体的成熟度划分和级别，以及与其相关的各个关键域的级别，定义了改进的先后顺序。虽然你还是在增量地进行过程改进，但是 TPI 模型规定你应该以一个特定的顺序进行过程改进。我先前已经举过这样的例子，例如如何从 0 级上升到 1 级，如何从 1 级上升到 2 级。

因此,和 TMM 一样,TPI 规定了过程改进的顺序。而且,TPI 还根据各种检查点的不同顺序,规定了每个过程域改进的顺序。也就是说,在上升到 B 级之前,必须满足 A 级的所有检查点,以此类推。因此如果您希望以一种自底向上的、轻度文档化的理念来进行测试过程改进,那么 TPI 就是适合您的。

同样的,对于 TPI 的哲学理念,或者说 TPI 式的过程改进顺序,有很多适用的情况。例如,一个制作大众市场 PC 软件或者网页的、处于起步阶段的公司,需要建立一个测试团队,并快速地引入一个过程,随后再稳定地、增量地进行改进,那么在这种情况下,应该使用 TPI 模型。<sup>6</sup>

## 8.7 用 CTP 改进测试过程

### 学习目标:

(K2) 总结 CTP 中定义的测试改进过程。

(K2) 论述测试改进模型 CTP 中的评估准则。

大概在 2000 年的时候我写过一本书“Critical Testing Process”,在本书的开头,我提出了一个假设,即有些测试过程是关键的,而有些则不是。我为测试过程改进设计了一个极度简要的方法,在这个方法中,测试团队和测试经理只需要关注一些他们必须正确执行的测试过程域。这个模型与 TPI 和 TMM 这种大成本的模型形成了鲜明的对比。

哪一个更好?显然,我倾向于使用一个专注的、极简的模型。不过,如果想要一个能给您完全指导的模型,而且您也不在意花点时间去关注一些并不是很重要的过程域,那么 TMM 或 TPI 模型会适合您。如果不喜欢这两个模型中的规范性的特性,可以忽略与成熟度模型相关的内容,只看与过程评估相关的内容。

让我们回到 CTP 模型,马上我就会讲到什么是关键测试过程模型。简而言之:如果一个测试团队能正确地实现关键测试过程模型,那么这个团队几乎总是能获得成功,不过如果它的实现很糟糕,那么即使这个团队里有天才的测试员和测试经理,它还是会失败。

我希望一个改进模型是专注的而又灵活的。因此,CTP 模型是允许裁剪的。它允许你识别并应对你的测试过程所面临的特殊挑战。它定量地、定性地识别了好的测试过程应该具备的各种特性。它允许你根据业务价值或组织的需要来确定改进的优先级。而且这个模型经过调整以后可以应用于所有软件开发生命周期模型。

我希望专注于一小部分过程,那么我是如何来选择的呢?什么是一个关键测试过程?

首先,我将过程定义为一系列操作、观察和决策。这个定义采用了一些 TMM 和 TPI 模型中的过程域,并将它们放到了更大的环境中去。例如,测试策略的制定成为测试环

---

<sup>6</sup> 要获得更多的信息,请阅读 Martin Pol 和 Tim Koomen 的“Test Process Improvement”一书,同时我还推荐 Koomen 等人写的一本书,叫做“T-Map Next”,因为 TPI 模型就是来源于 TMap 模型。



境建立过程的一部分，而办公环境则隐式地包含在测试系统开发过程这个过程中。然后，我将测试定义为计划、准备、执行以及优化的一系列活动，这些活动用于对一个系统的质量进行评估。

那么，在定义了什么是测试过程之后，我们如何确定一个测试过程是否是关键测试过程呢？有4个准则：

- 这个过程是否在不断重复，因此它影响了测试团队和项目团队的效率？
- 这个过程是否是高协作性的，涉及了大量的人员，尤其是各种不同类型的人员，影响到了测试团队和项目团队的团结与协作？
- 这个过程对于同行和管理人员是否是可见的，以至于影响到了测试团队的可信性？
- 这个过程是否影响到了项目的成功与否，例如影响项目团队或测试团队的有效性？

换句话说，一个关键过程直接而明显地影响着测试团队查找缺陷、建立信心、降低风险以及获取信息的能力。

根据上述准则，我总共识别了12个关键测试过程：

- 测试。这是一个总的过程，要从宏观的、策略的层次进行分析。它包含了11个子关键测试过程。
- 建立环境。这个过程明确了测试在项目和组织的定位。阐明了各方面对测试的期望。本过程为所有其他测试过程的裁剪建立了一个基础。在合适的时候，可以在这个过程中制定测试方针和测试策略。不过这两个文档并不是规定的，因为可能有些情况中我们需要，而在有些情况中则不需要。
- 质量风险分析。这个过程识别了系统质量的关键风险。它将测试与系统质量的关键风险联系在了一起。在这个过程中，质量和测试的利益相关者将就测试哪些项（到什么程度）以及不测试哪些项（为什么）达成共识。正如您所看到的，除了基于风险的测试策略之外，这个过程要求对策略进行调整。
- 测试估算。这个过程对测试的成本和时间以及项目的需要和风险进行了权衡。这个过程准确地预估计了测试的任务和持续时间。通过说明测试的投资回报，证明所需的测试工作量是合理的。
- 测试计划。这个过程使测试团队和项目团队成员达成共识，并做出承诺。它为所有的测试人员提供了一个详细的前进路线。同时还为项目回顾会议和未来的项目收集了所需信息。
- 测试团队发展。由于测试团队决定了测试的成败，因此这个过程将测试团队的技能与关键测试任务匹配在一起。它确保了团队具备解决关键问题的技能。这个过程持续地将团队的能力与测试的组织价值联系在一起。
- 测试系统开发。这个过程确保测试覆盖到了与系统质量相关的关键任务。设计测试以满足客户和用户对质量的要求。同时还对资源和时间的需求以及风险的关键程度进行了权衡。这个过程包括了测试用例、测试数据、测试规程、测试环境和其他辅助材料。

- 测试发布管理。如果没有测试对象，我们就没办法进行测试。如果测试项与测试环境不兼容，那么我们也无法进行测试。如果这次测试发布还没上次好，那么显然我们并未正在走向成功。因此，这个过程专注于如何将稳定的、可靠的测试发布安装到测试环境中。
- 测试执行。在这个过程中，包含了执行测试用例、将测试结果与预期结果进行比较、生成缺陷信息等活动。换句话说，这就是体现测试价值的时候。这个过程要花费大量的资源。通常在项目的结尾才会执行这个过程，然后就是项目的结束活动了。
- 缺陷报告。这个过程为改进系统的质量提供了机会（因此也降低了成本）。测试执行体现了测试的价值，而本过程则将测试的价值转移到了项目团队中，尤其是其他独立负责人以及基层经理。它提升了测试人员在开发人员心中的可信性。
- 结果报告。这个过程为管理层对项目做出决策提供了所需的数据。这个过程将测试的另一些价值转移到了项目团队中，尤其是基层经理、高级经理和主管。由于测试报告通常是坏消息，因此我们要将消息本身与提供消息的人区别开。它提升了测试人员在经理心中的可信性。
- 变更管理。这个过程允许测试团队和项目团队对他们所了解的信息做出响应。它正确排列了变更的顺序，将注意力集中在投资回报最高的活动上。

您应该已经注意到，对于每个过程，我都描述了其最优的状态。如果一个过程并没有满足这些能力标准，那么它就还不是最优的，还有改进的空间。

我们如何使用 CTP 进行评估和改进呢？若要使用 CTP 进行过程改进，那么在一开始应该先对当前的测试过程进行评估。在评估中我们要识别出 12 个测试过程中有哪些是没问题的，哪些是需要改进的。评估的结果是一系列按优先级排序的改进建议。评估人员的建议是基于组织的需要而不是某些成熟度模型。

具体如何进行评估可以根据组织的情况及其需要进行调整。我们可以只对一个测试团队进行 CTP 评估，也可以只对一两个测试过程进行 CTP 评估，比如测试系统开发，我们对各种影响质量的事物进行过 CTP 评估。CTP 评估是可以变化的，因此在 CTP 评估的过程中我们会测量下列度量元：

- 缺陷检测百分比。
- 测试投资回报。
- 需求覆盖和风险覆盖。
- 测试发布开销。
- 缺陷报告的退回比例。

除此之外，在 CTP 评估中我们还对下列定性因素进行评估：

- 测试团队角色和有效性。
- 测试计划的有用性。
- 测试团队的测试技能、领域知识和技术。
- 缺陷报告的价值。
- 测试结果报告的有用性。



### ■ 变更管理的效用和权衡。

一旦在评估的时候发现了一些可以改进的地方，评估人员就可以制定计划去实现这些改进。在 CTP 模型中包括了每个关键测试过程改进的指导方针，不过实际使用的时候，评估团队要对这些指导方针做很大的修改。

CTP 模型是非常灵活的。它假设测试主要使用一个分析式的测试策略，同时辅以动态测试策略。也可以将 CTP 模型调整成主要使用其他测试策略，比如基于检验表的策略或基于模型的策略，不过这么做的话对模型会有系统性的影响。如果我有机会写关键测试过程的第二版，那么也许我会进一步增加一些灵活性。

图 8-4 是我们为一个客户进行 CTP 评估的结果。您也许应该花一点时间阅读这个评估总结。

1. 测试团队和过程的价值：不错
2. 测试团队和组织的匹配协调：需要改进
3. 测试和质量的匹配协调：有改进的空间
4. 测试估算：有改进的空间
5. 测试计划：基本上很强，但回归测试策略需要加强
6. 测试团队发展：正在正面发展，优化一下方向
7. 测试系统开发：正在改进
8. 测试发布管理：优秀
9. 测试执行：正在改进
10. 缺陷报告：不一致；有改进空间
11. 测试结果报告：不完全，对很多利益相关者不合适
12. 更改管理和缺陷跟踪过程：需要改进

图 8-4 CTP 评估总结

正如你所看到的，我们的评估是很有条理的，与 TPI 非常类似，在这个评估报告中，我们并没有将过程域定为满足的或不满足的，而是给出了一个定性的评估。我们会特别指出一些可以改进的地方，例如一个过程的某些特定区域。评估并不仅是一个快照，同时它还能捕捉组织中某个特定过程域的趋势。

在图 8-5 中，可以看到一个测量测试系统开发过程的定量度量元。这个度量元关注测试用例细节。“测试用例细节”也可以叫做“测试规程”，是指测试人员执行手动测试的时候需要做的实际工作。

- 度量元：测试用例详细程度
- 使用：评估测试用例的详细程度
- 所需数据：每个用例的字数和执行用时（分钟）（仅统计手动测试），至少每 10 个用例所需的测试人员和应用
- 公式：字数/用分钟数
- 注意事项：
  - 样本越大越好
  - 与平均值和标准值的偏差

图 8-5 用 CTP 模型测量过程

这里我们评估的是测试用例的详细程度是否适合业务需求。CTP 要求的既不是轻度的文档化,也不是重度的文档化,而是测量当前文档化的实际状态,再将这个状态与业务需求放在一起进行评估。在这个度量元中,我们关注的是测试用例(或测试规程,取决于评估组织的命名规范)。

要计算这个度量元,我们要提取至少 10 个测试用例作为样本。我们计算每个测试用例中的字数以及执行这个用例所需要的时间。同时我们还计算它们的平均值和方差。在我们进行评估的时候经常会发现有一个申请提出要对用例进行大的改变,或者有时候甚至会有两三个这样的请求。如果这种改变是经过深思熟虑的、受控的,那么这是可以的,然而,通常情况下这都是由于缺少明确的管理以及项目成员不理解文档的什么地方需要详细一点、什么地方可以省略一点而造成的。

这个度量元还有其他一些需要注意的点。其一,我们要对各种测试人员和应用程序进行测量,样本的数目越大越好。另外,这个度量元只能用于手动测试,对于自动化测试用例我们使用其他的度量元。

每个关键测试过程都有约 5~10 个度量元,以及一些定性的评估。一般而言,我们不会在一次评估中对每个过程的每个度量元都进行计算,因为这些度量元与模型一样是可以调整的。因此,这是一种以客户为中心的评估。<sup>7</sup>

## 8.8 用 STEP 改进测试过程

### 学习目标:

- (K2) 总结 STEP 中定义的测试改进过程。
- (K2) 论述测试改进模型 STEP 中的评估准则。

系统化测试以及评估过程(STEP)模型与 CTP 一样,是一个非规范性的改进模型。也就是说,可以根据业务需要进行改进,而不是根据绝对的成熟度模型。STEP 对整体的测试方法做了一些假设,包括当前的方法以及组织期望采用的方法:

- 假设使用的是分析式的基于需求的测试策略。
- 假设在软件生命周期的早期就开始测试,同时测试使用一些测试过程模型。
- 假设测试被用作需求以及使用模型,也就是说在开发需求和用例的时候,测试对它们进行了确认,因此这要求并行开发系统需求规格说明和测试设计规格说明。
- 假设测试件的设计不仅与系统设计重叠,实际上更是在引导系统设计。
- 假设测试过程专注于早期的缺陷检测甚至缺陷预防。
- 假设使用了根本原因分析和其他复杂的技术,例如在第 3 章中讨论过的那些技术,这些技术系统地对缺陷进行了分析,以便在早期对缺陷进行预防和检测。
- 假设测试人员和开发人员能协同工作,完成所有任务。

正如你所看到的,上述的这些假设和需求与 ISTQB 大纲的原则和概念是一致的。这

---

<sup>7</sup> 我的“Critical Testing Processes”一书描述了关键测试过程模型。



些假设和需求使得 STEP 模型适用于某些组织。如果一个组织当前满足了其中一些假设,并且也赞同其他的假设,那么 STEP 就非常适合这个组织。即使这些假设没有一个满足的,但如果这个组织认同这些假设,那么 STEP 模型还是能够指导这个组织应该怎么做。不过,如果有些测试团队既不满足这些假设,而且它们所在的组织也不认同这些假设,那么,STEP 模型可能就无法满足组织的需要了。

那么,我们应该如何使用 STEP 模型进行评估和改进呢?模型中定义了需要评估的 3 个测试的主要阶段,包括:

- 计划。
- 信息获取,这包括两部分:测试分析、测试设计和实现。
- 测量,这包括两部分:测试执行、测试报告。

仔细看看,会发现这和 ISTQB 的基础测试过程有点类似,当然,在一些细节上还是有些区别的。

STEP 模型在这些阶段所用的文档是基于 IEEE 829 标准的。实际上,David Gelperin,STEP 模型的作者,也参与了 IEEE 829 标准的定义工作。因此,STEP 模型实际上是一个包装了 IEEE 829 文档标准,由一家叫做软件质量工程的美国公司开发的专有过程。

STEP 模型包括了定量的度量元和定性的会谈。下列这些是定量的度量元:

- 目前为止的测试状态。
- 测试需求覆盖或风险覆盖。
- 缺陷的趋势,包括检测、严重程度以及分布。
- 缺陷密度。
- 缺陷移除的有效性。
- 缺陷检测百分比。
- 缺陷引入、检测和移除的阶段。
- 测试的成本:时间、工作量以及资金投入。

下面这两个因素是定性的:

- 已定义的测试过程的效用。
- 客户满意度。

STEP 模型自身是非规范性的。我和我的同事曾与许多的客户一起,根据他们的组织需求,对 STEP 进行裁剪,然后投入实际的应用。

我曾提到过,STEP 模型假定测试主要使用的是分析性的测试策略,即基于需求的测试策略。同时,该模型还通过对需求元素进行评估来辅助测试。

与 CTP 一样,STEP 模型允许使用动态的测试策略。还有与 CTP 类似的是,可以对模型进行裁剪使其能应用于以其他测试策略为主的测试中,例如基于检验表的测试或基于模型的测试,不过这么做会对这个模型产生系统性的影响。

STEP 模型是非规范性的。测试团队可以用它来改进任何过程或任何过程的任何部分。不过,人们通常会将它和 TPI 混合使用以获得一个成熟度评级,这会使得 STEP 模型变成一个规范性模型。

## 8.9 能力成熟度模型集成 (CMMI)

### 学习目标:

(K2) 总结 CMMI 中过程域的验证和确认。

(K2) 论述 CMMI 中过程域的验证和确认。

在 20 世纪 80 年代,美国国防部要求卡内基梅隆大学的软件工程研究院,位于宾夕法尼亚州的匹兹堡市,开发一个能力成熟度模型。能力成熟度模型(或 CMM)依赖于全面质量管理(或 TQM)。而全面质量管理又是来源于由朱兰、戴明以及一些日本从业者提出的统计质量控制的概念。全面质量管理的核心理念是生产过程的质量很大程度上决定了最终产品的质量。这个理念在 CMM 模型、CMM 模型的变种和衍生模型的各个方面也都有所体现。

软件工程研究院对这个模型进行了拓展,生成了一系列变种,包括软件开发、系统开发、软件收购等。在 20 世纪 90 年代末 21 世纪初,软件工程研究院开发了能力成熟度模型集成(或 CMMI),该模型集成了上述这些变种。

能力成熟度模型集成是一个五级成熟度模型。这 5 个成熟度级别有:

- 初始级:过程是无法预期、无法控制的,只能被动地响应事件。
- 管理级:在项目层已经建立了过程,但是还是只能被动地响应事件。
- 定义级:在组织范围内已经建立了过程。这个过程通常能预料到事件的发生,并能对其进行控制。
- 定量管理级:在管理级别对过程进行了测量和控制。我们能收集数据并对比不同项目的数据。
- 优化级:不仅已经在组织级别对过程进行了测量和监控,比较不同项目的数据,同时强调使用这些数据以及其他信息去不断改进这个过程。

因此,可以看到其实 CMMI 和 CMM 是一致的。

那么,既然我们已经有了 CMMI,为什么还需要 TMM 以及其他一些测试过程模型呢?因为 CMM 和 CMMI 模型专注于过程改进,而对测试则不够重视。从理念上而言,软件工程研究院相信对产品质量而言一个优秀的过程是最重要的,而测试是次要的。对于生产线上的产品,这种理念是适用的,然而对于软件和系统,仅仅靠这个理念来保证质量是不够的。

CMM 与 TMM 有点类似,专注于阶段式的实现。而 CMMI 允许持续的实现或阶段式的实现。在持续的实现方法中,CMMI 为每个过程域定义了不同的成熟度等级。不过,实际上我很少看到有人以持续的实现方式来使用 CMMI 模型,因为许多组织只是将 CMMI 成熟度等级作为市场推广的一种手段而已。

与 TMM 类似,CMMI 的阶段式实现有 5 个成熟度等级。每一个级别都要求满足上一级别中的所有过程域。不过,在持续的实现中,组织可以根据自身需要,专注于对最重要的过程域进行改进,这个时候组织可以不用像阶段式实现中那样一定要满足所有作



为前提条件的过程域。

CMMI 中的阶段式实现确保了与 CMM 的通用性。而持续的实现则更为灵活。不过，在第 3 章讲外包测试的时候我曾提到过，很多组织将 CMM 或 CMMI 认证作为市场推广的一种手段。对于这些组织而言，阶段式的实现更加重要，因为这样他们就能宣称自己达到了某个级别。

在 CMMI 模型第三级定义级中，有两个过程域是特别值得我们测试人员注意的。第一个是验证。在基础级大纲以及 ISTQB 术语表中，对验证的定义是确保工作产品满足它们的需求。验证工作包括检查设计文档中是否体现了每个需求元素，是否每个设计元素都已实现等。

CMMI 为验证定义了 4 个主要活动：

- 选择要验证的工作产品。
- 建立验证环境。
- 建立验证规程和准则。
- 执行验证。

这些活动不是一次性的，而是嵌入在软件生命周期的各个部分。

第二个过程域是确认。同样的，在基础级大纲以及 ISTQB 术语表中，确认的目的是为了证明产品以及产品的组件满足了其预期的用途和环境。

CMMI 为确认定义了 4 个主要活动：

- 选择要确认的产品。
- 建立确认环境。
- 建立确认的规程和准则。
- 执行确认。

再次说明，这些活动不是一次性的，而是嵌入在软件生命周期的各个部分。

与确认和验证相关的活动包括了静态的和动态的测试过程，这与 ISTQB 基础级大纲和高级大纲也是一致的。换句话说，ISTQB 的软件测试概念与 CMMI 中的验证和确认过程域并不存在不一致的地方。不过，我在第 3 章中讲过，CMMI 中的过程域并不能满足测试实践的需要。因此，可以使用本章中讲过的测试过程模型，辅以 CMMI 模型来进行正确的测试。

我在本书中多次将 IVR 项目作为一个案例进行研究，现在我们将这个案例与 CMMI 联系起来。在这个项目中，有一名咨询顾问要评估 IVR 项目是否遵守 CMMI 模型。他检查的过程域之一便是确认。具体是怎么做的呢？

- 选择要确认的产品，我们对 IVR 的各个系统以及其综合系统进行了多个级别的测试。我们对每个主要的系统进行了正式的系统测试。我们还对这些系统进行了前置集成测试。同时所有系统都集成完毕之后，我们进行了正式的系统集成测试。
- 建立确认环境，在测试过程中我们不断增加测试环境与真实运行环境的相似程度。在每个系统测试级别中，其目标是要能模拟真实的运行环境。前置集成测试是在拟真的环境中进行的。系统集成测试是在完全真实的运行环境中进行的。
- 建立确认的规程和准则，我们在每个级别都定义了测试用例和退出准则。随着测

试级别的上升,每个测试的真实程度都在增加,以及每次入口准则和出口准则都会更加严格。

- 执行确认,我们按照计划和用例执行每个级别的测试。不过我们并不是总是想尽办法去满足入口和出口准则。

因此,虽然整个项目并不满足 CMMI 第 3 级,但我们的测试已经满足了第 3 级,即定义级。

## 8.10 测试改进过程练习

作为 HELLOCARMS 系统的测试经理,你想要在这个项目中对你们的测试过程进行改进。参考前面我们已经完成过的练习,回顾一下 HELLOCARMS 项目的背景。假设你目前处于第二个迭代的末期,项目还会持续约 5 个月。

概述将会对哪些测试过程进行过程改进以及如何进行测试过程改进。可以在我们讲过的 4 个测试过程模型中任选一个作为您的改进框架,或者也可以自己造一个。

如果目前是在教室里接受培训,那么大家可以分为 3~5 人一组。每一组给出一个方案,然后大家再一起讨论。

我建议用 20 分钟概述改进内容,然后用 10 分钟进行讨论。

## 8.11 测试改进过程练习参考答案

我将使用关键测试过程框架来概述我将做哪些改进。

### 1. 测试

由于在第二次迭代的末期我们关心的一个主要因素便是缺陷数目,因此我想要测量一下第一次迭代时的缺陷发现百分率。先选出第二次迭代中发现的 100 个缺陷样本,然后确定其中哪些在第一次迭代测试中就已经发现过了。这样我就能估算出有多少缺陷在第一次迭代中被遗漏掉了,以及第一次迭代的缺陷发现百分率。

根据这个分析的结果,我就能在后续的迭代中采取恰当的步骤来对测试检测缺陷的效率进行改进。

### 2. 建立环境

为了确认在上一个练习中列出的测试的商业价值,我会对测试的利益相关者进行调查,确认他们对测试价值的理解。如果我发现我的测试团队遗漏了某项价值,那么我会迅速将其补充到列表中。

我将会使用质量成本,基于估算的成本或历史数据,来量化测试的价值。然后我会尝试降低检查每一缺陷所用的成本,例如引入一些简单的测试自动化工具或使用测试外包,来进一步增加测试的量化价值。

我还会检查第一次迭代和第二次迭代中是否有一些回归程度很高的功能。如果存在的话,我会与开发经理进行协商,看看是否能用一些自动化的单元测试用具,如 JUnit



或类似的工具，以便在生命周期的早期就降低回归的风险。

总之，我会检查上游的那些质量控制活动，如需求和设计评审、代码评审、静态分析，以及单元测试等，看看是否做得不充分，导致在每次迭代的系统测试中有一大堆缺陷。因为这么多缺陷会导致无法在固定的时间内完成测试，也就降低了准时发布系统的可能性。

### 3. 质量风险分析

为了检验质量风险分析的正确性，我会对到目前为止的这些缺陷进行评估。对以下方面会尤为关注：

- 在我们评估认为不太可能发现缺陷的地方发现了大量的缺陷。
- 在我们认为会含有较多缺陷的地方只发现了少量缺陷。
- 在我们评估为低影响的地方发现了少量的低影响的缺陷，以及大量的高影响的缺陷。
- 在我们评估为高影响的地方发现了少量的高影响的缺陷，以及大量的低影响的缺陷。

根据这次评估，我会对质量风险分析进行修改，重新分配测试工作量，并重新排列测试用例的优先级。

### 4. 测试估计

我会用甘特图对实际的工作量和所有任务的完成时间与预估的工作量和完成时间进行分析。如果这其中存在的差异表明剩余的3个迭代所用的计划是不可行的，那么我会对估算进行调整并重新获取管理层的批准。

### 5. 测试计划

我会对到目前为止的测试计划进行分析，尤其是那些存在不可预期或无法管理的项目风险的区域。如果实际情况与计划的差别很大，可能后面3个迭代的计划也是不可行的，或测试还存在其他一些主要风险，那么我会对计划进行调整并重新获取管理层的批准。

### 6. 测试团队发展

我会为我的测试团队列一张技能列表（在下一章节我们会讲到）。如果在这次评估中发现团队技能中存在一些不足，那么对于那些在接下去3个迭代中马上要用到的技能，要立刻进行改进。

### 7. 测试系统开发

我将检查在第一次和第二次迭代中百分之多少的测试用例是被阻塞的。然后再调查这些阻塞是否是由于测试用例、测试工具或测试环境设计问题造成的。如果确实是上述问题造成了阻塞，我将会试着去改善这些问题，提高测试的流畅程度。

我还会不断评估3个方面的测试覆盖率：风险、需求和代码。我会检查与第一次迭代相比，第二次迭代的测试范围是否增大了很多。如果是的话，我会识别出造成这种增长的原因，相关的项目风险，然后制定计划以便在接下去的3个迭代中控制这种增长。

### 8. 测试发布管理

我会检查第二次迭代中发现的大量缺陷中是否有与构建（build）或安装相关的问题。

如果有,我会对这些问题的成本进行业务分析,再根据分析的结果向发布管理团队提出变更建议。

### 9. 测试执行

测试完成图以及测试小时图表明了第二次迭代中的测试执行的效率和有效性。我会检查缺陷的发现顺序是否与其优先级一致。理想情况中,应先发现优先级高的缺陷。如果事实确实是这样的,那么这表明质量风险分析中的优先级排序是正确的,而且测试团队执行测试时也没遇到大量阻塞。不过,如果测试团队发现缺陷的顺序与缺陷的优先级不一致,那么就要进行根本原因分析,看看为什么是这样的一个顺序,以及如何去进行改进。特别是在测试执行了一段时间,而缺陷发现率依旧很高的情况下,能先发现优先级高的缺陷就非常重要了。

### 10. 缺陷报告

如果有大量缺陷报告未起到应有的作用,那么我就会进行检查:

- 缺陷修复委员会拒绝了缺陷报告,因为这并不是缺陷,而是正确的功能。
- 缺陷报告被退回到测试团队,要求更多的信息。
- 开发团队关闭了缺陷报告,因为存在重复。
- 缺陷修复委员会推迟了缺陷报告。

我希望每种问题的缺陷报告数目不超过 5%。对于超过这个限度的问题,我会进行调查。

### 11. 结果报告

我会询问各种项目利益相关者,测试团队的报告是否是完整的、中肯的、及时的以及简明的。如果他们对测试结果报告有什么改进建议,那么我会接受这些建议。

### 12. 变更管理

我会检查是否在每次迭代中都会增加很多额外的特性,然后我会评估这些增加的工作量是否超出了我的测试团队的能力范围。如果是,那么我将会修改我的资源计划,并与管理层进行协调。

## 8.12 认证考试模拟题

在每个章节结束之前,您可以尝试回答以下样题,加强对所学内容的理解,并准备 ISTQB 高级测试经理认证考试。

1. 在下面哪个标准中可以找到可用的测试计划模板?
  - A IEEE 1044
  - B ISO 9126
  - C IEEE 829
  - D DO 178B
2. 对于过程改进而言,下面哪一项与回顾会议相关的活动最适合?
  - A 确保管理层承诺了支持进行改进
  - B 允许参与回顾会议的成员发表一些主观的看法和评估



- C 要求每个项目在测试结束过程中召开回顾会议  
D 禁止管理人员参与回顾会议
3. 假定您是一名可编程温控器项目的测试经理，该温控器用于控制中央加热、湿度以及空调系统（HVAC）。在发布之后，您发现质量风险分析未能发现某些重要的风险项，导致有些缺陷遗漏到了发布版本中。  
假设使用 TPI 模型改进测试过程。那么为了解决上述问题，您应该改进哪一个 TPI 过程域？  
A 生命周期模型  
B 测试方针和目标  
C 测试介入的时间  
D 估算和计划
4. 下列哪种说法更能精确地表达在大纲中讨论的 4 个过程改进模型中评估准则的区别？  
A TMM 和 TPI 用成熟度模型来指导改进，而 CTP 和 STEP 则提供评估改进的有效性的方法  
B CTP 和 STEP 用成熟度模型来指导改进，而 TMM 和 TPI 则提供评估改进的有效性的方法  
C CTP 和 TMM 用成熟度模型来指导改进，而 CPI 和 STEP 则提供评估改进的有效性的方法  
D TMM 和 STEP 用成熟度模型来指导改进，而 CTP 和 TPI 则提供评估改进的有效性的方法

## 第9章

# 测试工具和自动化

“什么是用于自动化的商业案例？”

——作者问一个测试团队，该团队已经为某客户服务了四年，但却没有明显提高测试用例自动化的百分比。他们的回答不着边际。

高级大纲的第9章是测试工具和自动化，与基础级大纲中的内容相比，高级大纲中增加了一些概念性的知识。另外，高级大纲还对工具的分类进行了详尽的描述。本章共包括3个部分：

1. 概述。
2. 测试工具概念。
3. 测试工具分类。

下面一一讲解这些小节，看看它们与测试管理有什么关联。

## 9.1 概 述

### 学习目标：

能回想起该部分内容即可。

本章将对基础级大纲中描述的工具相关的基本概念进行拓展。我们先讲解一些工具概念，然后是具体的工具。

所有测试人员都应该对测试工具有一个基本的了解，知道应该用什么工具，不应该用什么工具。很多时候，组织和个人对测试工具抱有不太实际的期望，他们认为，只要选择了正确的工具，就能解决一切测试问题。这是不现实的。

在高级大纲中，我们根据角色对工具进行了分组，例如，测试经理、测试分析师和技术测试分析师。当然，有些工具的使用范围更广一点，适合多个角色使用。

## 9.2 测试工具概念

### 学习目标：

(K2) 从下列多种角度对不同类型的测试工具的概念进行比较：益处和风险、测试



工具使用策略、工具集成、自动化语言、测试准则、工具部署、开源工具、工具开发和工具分类。

(K2) 理解测试工具部署的不同阶段。

(K2) 描述为什么/何时需要创建测试工具使用策略或工具使用的路线图。

测试工具是非常有用的。事实上，有一些工具是必须的。很难想象一个包含两个人或更多人的测试项目却没有一个事件跟踪系统。通常情况下，测试工具能提高测试的效率和精确性。不过，只有当您仔细地选择并部署了工具，才能享受到工具带来的好处。

一般我们将测试自动化的理解为测试执行的自动化，而实际上我们也能将测试过程的其他过程自动化。大多数的测试自动化是用于自动化那些乏味的工作，例如回归测试，或难以手动执行的工作，例如性能测试。

若要充分利用测试工具，那么除了要仔细地选择和部署工具外，还需要对其进行不断地管理。您应该制定计划，对测试工具、测试脚本、测试数据和其他的测试工具项进行配置管理，以及将测试和测试工具的版本号与测试项的版本号联系在一起。

您应该为自己的测试自动化系统制定一个恰当的架构。身为一个咨询和实践者，我经常看到一些测试团队由于在开始自动化测试的时候采取了错误的决策而遭受了挫折。

一个好的架构能对自动化的另一个重要方面产生作用，即建立和维护测试库。在对测试用例的大小、测试用例和测试数据的命名规范，以及与测试环境的交互进行了统一之后，就可以使用工具建立可重用的测试模块。应该为这些测试模块建立一个库以便存储。

自动化测试本身就是一个项目。因此，类似于其他的项目，您需要根据项目的复杂度以及所需时间来制定一份文档，包括自动化测试如何进行、为什么要这样进行测试等。这份文档不必太花哨，不过大多数自动化测试系统都应该具备这样一份文档。

记住，应该在计划中考虑到今后可能的拓展和维护。如果不提前考虑，尤其是如何维护测试，那么极有可能导致测试自动化失败。

最后，如果想要使用测试执行工具，还必须拥有一份测试准则。测试准则用于确定预期结果。对于自动化测试而言，需要一些可用的测试准则而不是依靠人为判断。

这里我举个例子说明一下。一个计算器项目需要另外一个计算器作为测试准则。在这个例子中，可以在 Excel 表格中预先定义好测试及其预期结果，并作为测试执行时的参考。而对于有复杂业务逻辑的程序而言，如果想要一个完全自动化的准则，那么必须具备一个与被测软件一样的程序，不过通常是不可能存在这样的系统的，除非您是在更新一个遗留系统。在这种情况下，先要使用被测软件生成输出，然后更新那些需要验证的地方。这类准则被称为基线，您可以使用基础级大纲中提到的比较器对实际结果和基线进行比较。

### ISTQB 术语

**测试准则：**在测试时确定预期结果与实际结果进行比较的源。一个准则可能是现有的系统（用作基准），一份用户手册，或者是个人的专业知识，但不可以是代码。

如果存在测试准则，但是更新不及时的情况下，基线的这个概念还是很有用的。

如果要进行的是可用性测试，记住，这种测试需要人的直觉去体验测试的结果。因此，自动化的可用性测试是没有有用的测试准则的，对这类测试进行自动化通常是在浪费时间。

上面我们简要地探讨了一下影响测试自动化的各个要点。现在我们将从商业用途的角度来深入讨论这些要点。记住，只有当测试自动化能带来明显的收益时才考虑使用它，包括能缩短测试执行时间、降低测试的整体成本或能覆盖一些特殊的质量风险。

我们所说的这些收益是与手动测试的时间、工作量和覆盖相比较而言的。只有与其他方法进行比较，才能评估出一种方法的投资回报。

### 9.2.1 测试自动化的成本

在任何商业案例中，我们都要考虑成本、风险和收益。先从成本开始，成本可以分为两种，初始成本和后续成本。初始成本包括以下方面：

- 评估并选择正确的工具。许多公司试图跳过这一步，然而最终他们会为此付出代价，因此要抵挡住这种诱惑。
- 购买工具或采用一个开源工具，或自己开发一个工具。
- 学习如何正确使用工具。包括组织内知识的传递和学习的成本，例如设计和记录测试自动化机构的成本。
- 将工具与已有的测试过程、其他测试工具和团队集成起来。

后续成本包括：

- 维护工具和测试脚本。测试脚本的持久性，即一个脚本在更新之前能使用的时间，是一个大问题。确保你设计的测试系统架构能将这个成本降到最低，换句话说，就是要最大化测试的持久性。
- 后续的许可证费用。
- 工具的支持费用。
- 后续的培训成本。例如，新员工入职培训以及工具更新后的相关培训。
- 将测试移植到一个新的平台上。
- 覆盖新的特性和应用程序。
- 处理与工具相关的问题，如可用性、限制和依赖等。
- 对你的测试脚本进行不断的质量改进。人们很可能会忽略这一步，但如果有不同的团队在进行测试自动化，而这些团队之间工具的使用方法和脚本互相不兼容的话，那么重用的可能性就大大降低了。我遇到过一个客户损失了 25 万美元，还未能按时发布产品，仅仅是因为两个自动化人员在各自开发一个类似的工具。

基础级大纲建议使用一个试点项目来引入自动化。这是一个好主意。不过，通常试点项目会遗漏一些重要的方面，如后续成本，尤其是维护成本。

我们也可以将成本分为固定成本和可变成本。固定成本是指无论有多少测试用例需要自动化我们都要付出的成本。主要的固定成本有购买工具、培训以及许可证费用等。可变成本取决于我们要自动化测试的数目。主要的可变成本有测试脚本开发、测试数据



开发等。

### 9.2.2 测试自动化的风险

除了成本，还需要考虑风险。基础级大纲中列出了一些风险。让我们再来看看：

- 对工具不切实际的期望。这是一个非常常见的风险，在那些没有多少自动化测试经验的组织中尤其常见。软件工程是一件很困难的工作，包括软件过程中的测试部分，为此人们总是试图寻找一些万能药、巫术来简化这项工作。这种愿望会导致人们产生一种幻想，认为确实存在一些新的过程方法、新的编程语言、新的管理技术，或新的工具，能够神奇地将一些复杂的问题变得简单。测试工具做不到这一点。所以应确保人们对工具有合理的期望，包括工具的功能和易用性。
- 低估了引入工具所需的时间、成本以及工作量。有些人看到在供应商给出的示例中，一个销售工程师在 30 分钟内测试了数十个界面或几百万行代码，就认为这就是工具的效果。然而，一个经过精心准备和包装的示例与在整个组织中推广一个工具是完全不同的概念。你除了要花大量的时间去建立测试框架、与其他工具集成、为工具的使用制定标准和指南等，还需要对工具的使用者进行培训，包括外部培训或外部咨询等。
- 低估了取得收益所需的时间和工作量。我的大部分客户花费了数年时间才在测试投资上获得了收益。而这些获得收益的客户还算是幸运的，因为我们看到过很多最终亏损的投资。测试自动化投资的收益通常包括以下几个方面：降低了整体的测试工作量、降低了测试执行的时间、覆盖了以前未覆盖的质量风险。测试自动化投资的收益是明显的、持续的。这种收益有时候还包括测试过程方面的改进以及工具使用方面的改进。
- 低估了维护测试所需的工作量。这个风险主要来自于两方面的因素。在很多时候，组织会减少初始的测试开发的投入以及工具引入的投入，创建一些脆弱的、难以维护的测试件。待会我在讲到高级大纲中提出的风险时会仔细讲解这一点。然而，即使一些组织不存在这个问题，他们还是会发现另一个问题，他们无法恰当地预估测试维护的预算。这将导致测试产生不正确的结果、降低测试自动化的投资回报，以及使得项目进入一个死循环。
- 过渡依赖工具。对于自动化，人们存在一个普遍的错误认识，他们认为给孩子一把锤子，那么所有事情就变成了钉子。实际上，情况要复杂得多，例如人们试图自动化一些无法自动化的测试。还有一种情况是人们没有测试那些无法自动化的区域，导致工具驱动测试覆盖存在缺口。当然，这是一种很荒谬的结果，因为测试覆盖决策不应该取决于测试工具以及工具的功能。

除了基础级大纲中列出的这些风险，还有一些其他风险需要考虑：

- 已有的手动测试可能是不完整的或错误的。在这种情况下，如果将手动测试用例作为自动化测试的依据，那么只是在让错误的事情做得快一点而已。在自动化之前，应该仔细检查手动测试用例、数据和脚本，因为稍后再要修改成本就会高很多。

- 创建的测试脚本、测试框架以及测试数据脆弱、难以维护、经常需要更新。这是一个典型的自动化测试风险。不过通过仔细地设计可维护的、健壮的、模块化的测试自动化框架、脚本和数据，或者使用其他一些技术，可以降低这个问题发生的可能性。如果在工具引入阶段发生了一些疏漏，那么我保证这些疏漏将导致测试自动化项目的失败。随后的测试维护工作将会迅速消耗用于测试自动化的资源，导致自动化测试停顿不前。我们发现一些组织直到测试自动化项目开始的数年后，才发现他们在测试框架、测试标准和指导，以及测试过程的设计中忽略了维护性问题。这些糟糕的设计导致了测试维护的成本不断升高，使项目进入死胡同。测试自动化的投资发生了亏损，以前的付出也变成了无用功。
- 您未能有效地监控测试自动化的投资回报，不断地在往一个无底洞内投钱。事实上，作为一名测试经理，这是您所面临的最糟糕的局面。迟早有人会发现项目在不断亏损。这个时候，其他经理包括领导将会质疑您的管理能力。因此，如果您无法处理或扭转这种不利的局面，那么这种测试自动化方面的风险将会变成您的工作和职业生涯的风险。
- 由于每个人都专注于运行那些脚本化的、不变的、没有人为参与的自动化测试，导致缺陷检测的有效性大大降低。自动化测试在建立信心、管理回归测试以及重复测试方面做得都很好。但是，在手动测试的时候人们还会做一些自然而然的探索，而自动化测试却做不到这一点。你应该确保在你的测试中包含了适当的手动测试。

正如您所看到的，应该对上述这些风险进行管理。如果测试自动化能解决问题，那么为什么不用呢？

### 9.2.3 测试自动化的益处

我们支付了上述这些成本，承担了一系列风险，就是为了获得相应的收益。那么测试自动化的收益有哪些呢？

首先，我必须强调一下，聪明的测试团队对测试自动化都有一个明确的目标，通过创建可重复的、低维护成本的自动化测试，他们将在数月或数年内获得回报，因此他们会在开发自动化的测试用例、测试数据、测试脚本和其他自动化项目上投入大量的人力、物力去实现这个目标。我所说的大量投资是指，这些聪明的测试团队不会削减初始阶段的测试自动化开发的投入，因为他们明白这将降低今后的收益。

聪明的测试团队会仔细地选择那些能够带来回报的测试用例进行自动化。不加思考地将所有已有的手动测试用例自动化只会带来亏损。

经过仔细地挑选以及精心地设计之后的自动化测试变得省时省力。由于测试执行的成本和持续时间大大降低，因此我们可以随心所欲地执行测试、提高整体覆盖率以及对发布版本的信心。也就是说，在投入相同的情况下，能够进行更多的风险缓解活动。每一个风险缓解的级别所需的工作量降低了，这样那些多余的精力就可以用于扩大测试覆盖（在不影响项目进度的情况下）。

如果测试执行所需的时间主要是由执行测试用例所需的时间决定的（与由修复缺陷



所需的时间相对),那么自动化能降低测试执行时间,提早发布软件。不过在推销这一益处之前,需要先检查一下,因为也许缺陷查找/修复/确认测试的这个周期的所需时间才是测试执行时间的决定性因素。

另外,有些测试类型无法通过手动测试执行,例如性能测试和可靠性测试。而有了自动化测试之后,我们就能覆盖到这些测试类型,降低风险。

在我们为自动化测试的这么多益处欣喜若狂之前,我还是要强调一遍我刚才提到过的一点。根据初始投入的大小,您可能要等上数月甚至数年才能使收益等于投入。要明白:在大多数情况下,这是没有捷径的。如果您试图降低引入自动化时的初始成本,那么可能会造成一种零回报甚至是亏损的局面。自己算算在这种情况下需要多少时间才能盈亏平衡您就知道这么做是否合算了。

那么,除了节省时间、降低工作量,以及更高的测试覆盖(或风险更低)之外,测试自动化还能带来什么益处?一方面,测试执行时间将更可预测。如果我们能在下班之前运行自动化测试,到第二天早上上班的时候发现测试已经全部执行完毕,这种感觉将是多么美妙啊,而管理层尤其喜欢这种感觉。另一方面,这种快速地执行回归测试和确认测试的能力还有一个附带的益处。由于我们能更好更快地管理产品中的变更和风险,因此我们在项目后期进行变更的能力也增强了。无疑,这是一把双刃剑,因为它可能会导致一些鲁莽的变更,不过小心使用这种能力,它能帮我们处理一些紧急变更。

由于与手动测试相比,自动化测试更具挑战性,也更受尊重,因此测试人员以及测试团队会发现自动化测试是值得做的。

最后,由于某些生命周期模型,尤其是敏捷模型和迭代模型,通常会有很多晚期变更,因此可以使用自动化测试,在不增加工作量的情况下管理回归的风险。

## 9.2.4 测试自动化的策略

我和客户在很多场合都说过,测试工具本身并不是一个测试自动化策略。现在举一个非软件的例子。我居住在得克萨斯州中部,在这里,有一望无际的杜松林,由于某些原因,人们称其为雪松。但它们并不是雪松,雪松高贵、正直、使用价值高。而杜松只能用于酿造杜松子酒而已,在冬天杜松开始授粉的时候,还会给得克萨斯中部三、四百万的居民带来困扰。

我家有11英亩(大约4公顷)的土地。我们周围到处都是杜松。我和我的妻子想方设法要摆脱它们。不过,去家得宝买一把链锯并不是清除11英亩杜松(有一些杜松有10~15英尺高)的策略。链锯只是一种方法,可以用来清除11英亩的杜松,而策略则包括了从哪里开始清除、清除后如何保证它们不再生长、如何处理这些锯倒的杜松等。

下面是测试自动化的一些策略。第一点也是最重要的一点,就是要根据长远目标进行自动化,建立一个可维护的自动化测试系统。只对那些可以自动化的测试用例进行自动化。自动化的测试在执行过程中不需要人为地判断结果。

对那些手动执行错误率很高的测试和任务进行自动化。这不仅包括了回归测试(这是一个非常值得自动化的目标),还包括了创建并载入测试数据。

只对那些存在业务用例的测试套件或测试用例进行自动化,这意味着你必须清楚从

现在开始到程序退役的这段时间内这个测试用例会被执行多少次。

即使大多数的自动化测试需要深思熟虑，要做大量的工作，但是还是可能找到并利用一些简便的方法，先摘那些低垂的果子。例如，如果您发现通过使用一个免费的脚本语言以某种特定方式运行程序可以会发现一些可靠性问题，那么就用吧！

但是必须指出的是，必须小心地管理测试工具列表，包括免费软件和商业软件。因为一不小心就会发现您的团队所使用的这些工具是无法管理的或者还未充分理解如何使用这些工具，尤其是当每个人都下载了自己喜欢的免费工具的时候最容易发生这种问题。制定一个测试工具评估和选择的过程并遵守这个过程。

为了维护自动化时的可重用性以及合规性，确保已明确定义了应该如何获取工具、如何选择测试进行自动化、如何编写可维护的脚本等。

如果在为安全关键系统进行测试自动化，那么就需要遵守不同的规则。例如，我曾为美国盟国的一个主要防卫设施供应商提供咨询，我和我的客户识别了一些需求，这些需求将会影响单元测试中自动化测试工具的使用：

- 这个工具必须支持和遵守（或至少兼容）DO-178B 标准。
- 这个工具必须能够面对一个事实，即目标系统与主机系统是不同的，一个是飞机上的嵌入式系统，而另一个只是用于编程的 PC。目标系统与主机系统之间的不同导致了一系列测试和开发相关的问题。
- 这个工具必须能够在一个充满了中断（interrupt）的环境中使用，并能够测试中断处理。
- 这个工具必须能够帮助我们管理创建现实的测试环境中遇到的困难（或至少不使情况恶化）。

我们还有一些客户也是安全关键领域的，其中有一些受到美国食品和药物管理局的监管。这些受到监管的客户所使用的测试方法和测试工具都受到了严格的限制，这些限制略有不同，但大致还是相似的。

### 9.2.5 测试自动化策略的案例分析

几年前，我们为一位客户的测试团队进行了评估，该团队在自动化方面遇到了一些困难。他们自己并没有意识到问题有多严重，不过至少他们发现相对于测试自动化的投资，他们的回报显得非常糟糕。在我的评估报告中，指出了基本的两点：

- 首先，我告诉他们当前所使用的测试自动化方法的效果有限。负载测试有点作用，不过受限于测试环境相关的问题。功能相关的回归测试一直处于无尽的维护中，然而在其他 99% 的与数据相关的应用却没有得到自动化覆盖，存在很大的风险。
- 第二，我告诉他们，尽管他们在前 4 年中在测试自动化方面投入一大笔钱（数百万美元），但是他们并没享受到任何回报。唯一一个获得回报的是那个外包公司，这个外包公司设计了一个不可维护的自动化测试框架，而现在这个公司就可以按小时对测试维护进行收费了。

由于存在回归风险，因此不考虑放弃自动化。我的建议是重新按照下面的这个路线图进行测试自动化，如图 9-1 所示。接下来我将从右上角开始一一讲解。



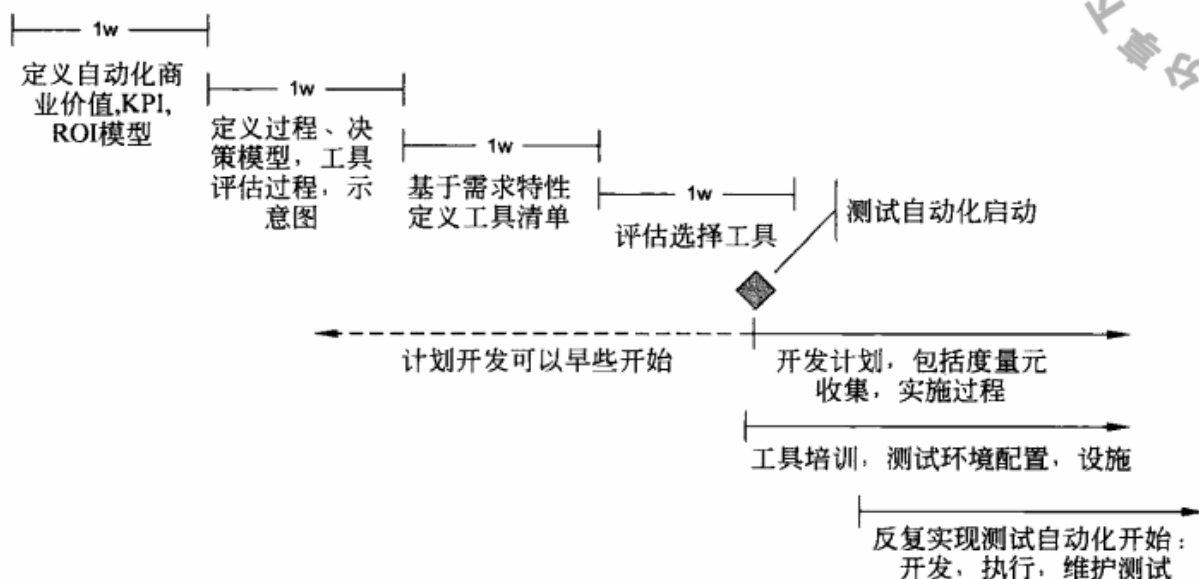


图 9-1 测试实现路线图

在第一周中，定义自动化的预期回报，选择自动化的关键过程指标，定义投资回报的计算模型。

在第二周中，定义选择测试进行自动化的过程，工具评估过程以及测量自动化收益的一览表。

在第三周中，制定一张候选工具列表，并将工具所具备的特性与所需特性进行对比。

在第四周中，使用定义好的、灵活的、现实的工具评估过程选择恰当的工具。

在这个时候，我会告诉他们，可以考虑重新开始测试自动化工作了。现在已经有了一个明确的策略来进行自动化，确保了能获得现实的、可测量的商业价值。

在第五周中，重新开始测试自动化工作。制定一个详细的项目计划（注意：我在图中已经标明这项活动可以提前到第二周）。制定测试自动化度量的收集机制。定义自动化的执行过程以及将自动化集成到上层测试过程中的过程。

还是在第五周，开始使用工具。为自动化测试工程师进行工具培训（我在报告中明确指出他们应该选择有相关知识背景的、使用过该工具的外包员工来进行测试自动化，这是他们之前失败的原因之一）。配置自动化的测试执行环境，创建自动化所需的其他基础设施。

第五周之后就可以开始创建自动化测试了，即开发、执行和维护测试。

### 9.2.6 测试工具集成和脚本

在复杂的测试自动化项目中需要用到测试工具。稍后我会举例说明这是什么意思。

许多组织会使用各种测试工具和开发工具。我们可能有一个静态分析和单元测试工具、一个测试结果报告工具、一个测试数据工具、一个配置管理工具、一个事件管理工具，以及一个有图形用户界面的测试执行工具。在这种情况下，如果能将所有测试结果集成到我们的测试管理工具中去，并增加测试与需求或风险之间的关联就好了。因此，你应该试着将工具集成起来使它们能进行信息交换。

您买了同一个供应商的测试工具套件并不意味着这个套件中的工具能集成在一起，

虽然它们应该是能集成的。不应该买这种不能集成的工具套件。

如果您买的工具无法完全集成在一起,那么可能不得不自行集成。在这么做之前,应先权衡一下手动移动信息的收益以及成本和风险。

近来,集成后的开发环境带来很多益处。我相信在不久的将来我们测试人员也将拥有集成的测试工具。

大多数测试自动化工具(至少是那些执行工具)能使用脚本语言。通常,我们会用脚本语言来写测试框架,然后用平面文件、XML 文件或数据库中的数据和关键字来驱动测试。脚本与数据的分离提供了可维护性。

在测试分析师和技术测试分析师这两本书中,我们花了很长的篇幅来讨论组合测试的问题。手动测试无法很好地完成组合测试,因为这其中的工作量太大了。不过,通过使用脚本语言,我们就能覆盖更多的组合。

有些工具能直接访问应用程序的 API。例如,有些测试工具能直接使用 HTTP 和 HTTPS 协议与网络服务器交互,而无需使用浏览器。

不同脚本语言的能力相差很大。有些脚本语言与通用编程语言类似。有些脚本语言是特定于某些领域的,如 TTCN-3。有些并非特定领域的脚本语言由于其特性因此在某些领域受到欢迎,如 TCL 在电话和嵌入式系统中比较常用。

并不是所有的工具都是要花钱的(至少无需购买)。有些可能是从网上下载的,而有些是自己编写的。现在存在很多开源的测试工具,和商业软件一样,这些工具的质量差别明显。我使用过一些质量很高的开源测试工具,也听有人说过某些工具很不好用。

即使开源工具无需购买,但是依旧需要花费时间和精力去学习、使用以及维护。所以与商业工具一样,需要根据被测系统的特性对开源工具进行严格的评估,而不只是执行一下包装过的样例。记住,包装过的样例几乎总是能正确执行的,除了基本的平台兼容性,这种样例表示不了什么。

除了质量方面的因素外,许可证也是需要注意的,在目前的开源工具中存在着不同类型的许可证,例如创意共享许可证和 GPL 许可证,您可能不得不公开对开源工具的修改。你们公司的管理层会在意是否会发生这种情况。作为一名测试经理,在团队内推广开源工具之前,应该坚持让高层或公司律师先审查一下许可证。

如果未能找到适合的开源工具或商业工具,可以自行编写一个。很多人都这么做。不过,这种方法的成本很高。而且工具的开发工作大多数情况下是作为一项兼职工作进行,因此存在很高的风险,一旦工具开发人员离职,这项工作就完全停止了。确保在工具开发过程中做好了文档化工作。

最后,警告:在测试安全关键系统时,可能存在一些监管相关的需求,这些需求规定测试工具必须通过哪些认证,影响到了对开源工具和自定义工具的使用。

### 9.2.7 集成化测试工具的案例分析

图 9-2 是一个集成化测试系统的架构。该自动化测试系统目前正用于测试一个保险系统,这个被测系统(或者更恰当地说,被测的综合系统)见图 9-2 中央。



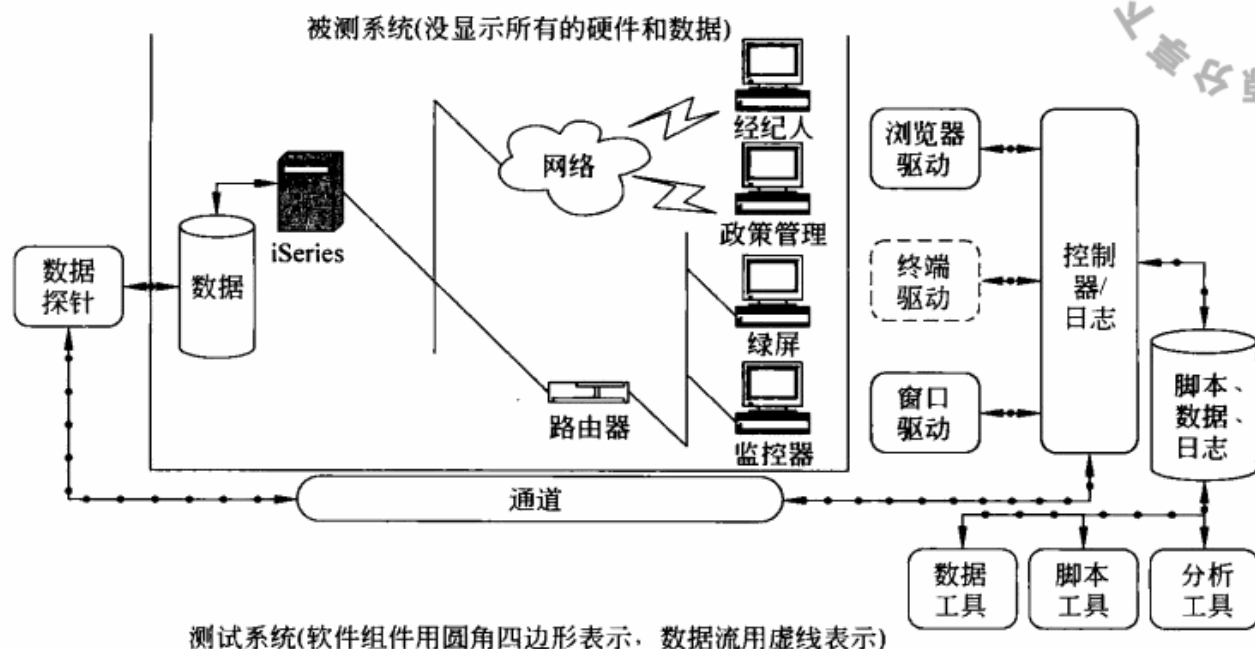


图 9-2 集成化的测试框架

被测系统的前端有 3 种主要接口类型：浏览器、遗留的基于 UNIX 的绿屏程序，以及一个新的 Windows 界面程序。前端应用程序通过保险公司的内部网和因特网与后端的 iSeries 服务器连接。iSeries 服务器管理着大量的信息，包括客户、保险合同、理赔历史、应付账款、应收账款等。

在图 9-2 的右边是测试自动化系统的主要元素。对于上述提到的任意一种接口类型，我们都需要有一个驱动与其对应，以便提交输入并观察输出。终端驱动的边框是用点划线表示的，因为目前还不能确定这个驱动是否有必要。控制器/日志模块根据已有的脚本库调用相应的驱动执行测试，并记录测试的结果。测试数据和脚本是通过工具创建的，同时该测试系统还存在一个测试日志分析工具。

图 9-2 右边的这些元素可以集成在一个工具中。不过由于这是一个测试系统的设计图，因此我们不讨论实现的细节。应该先设计测试，然后再选择合适的工具，而不是让工具来左右你的测试设计。相信我，我在这个地方摔倒过。如果让工具来驱动测试，那么最后会发现您没在测试那些重要的部分。

现在让我们来看看图 9-2 的左边和下方。在大多数复杂的应用系统中，界面操作只是很小的一部分而已。真正重要的是数据的传输、存储、删除等操作。因此，要想知道一个测试是否通过，我们需要检查数据。数据探针能帮我们做到这一点。

在数据探针和控制器之间的通道是用于传递请求和返回结果的一个组件。例如，如果某个特定操作需要向表中插入 100 条记录，那么控制器就会通过某个应用程序来执行这个操作（比如通过 Windows 驱动器调用 Windows 程序），然后再用数据探针检查这 100 条记录是否已正确添加。因为在屏幕上返回的消息表明操作成功，但实际上可能只插入了 90 条记录。因此我们需要有工具来查找这类缺陷，即数据探针。

现实中，右侧的这些工具通常会是一两个集成化的商业工具或免费工具，而数据探针和通道一般是自己开发的工具。

## 9.2.8 测试工具分类

工具有多种分类方法：

- 在基础级大纲中，我们根据工具支持的测试活动对其进行分类。
- 我们也可以根据工具支持的测试级别进行分类。例如，单元测试工具、集成测试工具、系统测试工具。考虑到多数工具支持多个级别，因此这种分类方式并不常用。
- 我们可以根据查找的缺陷类型对工具进行分类。
- 我们可以根据测试技术类型对工具进行分类。例如，在测试分析师那一册中可以看到成对测试工具和分类树工具。
- 我们可以根据使用目的进行分类。
- 我们可以根据工具测试的应用领域进行分类，当然，这种分类方式对特定领域的测试工具比较有用。
- 我们可以根据使用方式进行分类。
- 最后，我们还可以根据用户类别进行分类。在下一节“测试工具种类”<sup>1</sup>中我们就是用的这种分类方式。

在一开始部署自动化测试工具的时候，很容易犯一些愚蠢的错误，最终导致失败。接下来让我们来看看在自动化工具部署的关键阶段，应该考虑哪些重要因素。

很重要的一点是，所使用的自动化测试工具也是一种软件，而且通常都是比较复杂的。您可能用这个工具来解决一个复杂的问题，因此，解决方案应该能够持续使用，甚至能进行拓展。在用工具搭建测试系统的架构时应先制定好计划，以及在使用工具进行测试前也应制定相应的计划。

工具通常都有软件和硬件方面的依赖。有些工具能很方便地与其他工具集成，而有些则是单独使用的，在评估工具的时候应该考虑这个因素。

对于商业工具，一般对非标准平台的支持都做得不太好。因此，如果系统需要支持除了 Windows PC 或基于浏览器的应用程序之外的平台，那么您必须仔细考虑这个问题。有时候，您可能发现必须使用一个开源工具或自己开发一个工具。修改商业工具以期望它能适应某个环境通常都是不可行的，供应商能提供的支持（即使存在）也很有限。

当您手上有一个工具时，可能想要自动化每一个手动测试用例；通常来说这并不是一个好主意。您可以专门设计一些测试用例用于自动化，不过一般还是需要做一些修改，如格式化测试用例、考虑重用模式、拓展输入、使用变量而不是硬编码值来增加可维护性，以及设法充分利用工具的优点。测试工具可以浏览测试集、重复测试，以及改变测试的顺序，为我们提供了一些新的测试实现方法。当然，工具应该提供一些分析和报告功能，你会用到这些功能的。

自动化测试工程需要编程技术，但是可以编写一些自动化的测试框架让非程序员使

---

<sup>1</sup> 在该小节讨论工具和分类的时候，记住高级大纲只是在对基础级大纲进行更新和拓展。因此，你需要同时参考基础级大纲和高级大纲以便获得完整的信息。



用。编写一个大而复杂的测试框架是一项大而复杂的任务。如果没做好，那么你就会遇到我所说的零回报的糟糕局面。测试自动化人员，尤其是那些编写自动化框架的人员，需要接受测试工具、编程和设计方面的培训。

测试自动化并不是一成不变的，因为被测系统本身是在不断变更的。因此，自动化测试能正确进行并不代表你的测试结果有意义。一定要对自动化测试进行审计，包括手动执行一些回归测试用例，看看测试是否能正确执行以及对结果进行验证等。

另一个容易犯的错误是误以为现在做的事情就是今后将做的事情。然而实际上需求是变化的，一部分是因为被测系统的变更，另外是对工具的理解和使用也在发生着变化。因此，要考虑工具的可拓展性。一个有应用程序编程接口（APIs）、标准的脚本语言以及能与其他自动化工具交互的工具拓展起来就会方便很多。

最后，在选择工具的时候，无论选择的是商业工具、开源工具或自行开发一个工具，都必须对工具的能力进行仔细地评估。以下是应该考虑的一些因素：

- 该工具对测试分析能起到什么作用？
- 该工具对测试设计和实现能起到什么作用？必须手动设计测试，或者工具可以自动生成测试？
- 该工具对测试选择能起到什么作用？测试选择过程是纯手动进行的，或者工具可以根据某些准则自动选择测试？
- 该工具对测试执行能起到什么作用？测试执行是纯手动的还是自动化的（大多数测试执行工具能自动执行测试，而有一些测试设计工具无法做到这一点，它们只能通过接口与其他自动化测试执行工具交互）？如果测试是自动化执行的，那么执行前需要做多少准备工作？在自动化执行过程中是否需要人为干涉，如果是，那么为什么需要干涉以及什么时候进行干涉？当遇到常见的错误时，测试是否能自动重新执行？
- 该工具对测试评估能起到什么作用？若要使用测试执行工具，就必须有一个测试准则。那么对于评估、分析和结果报告你还需要什么？

对工具进行仔细地选择能避免在以后的数月、数年的自动化项目中遇到无尽的痛苦。

## 9.3 测试工具种类

### 学习目标：

（K2）从测试目标、易用性、优点、风险和用途等角度对不同测试工具进行汇总。

（K2）总结在测试安全关键系统中使用测试工具和开源测试工具的特别需求。

（K2）总结不同测试工具的特性和运用效果，总结不同测试工具部署、使用以及对测试过程的影响。

（K2）描述何时以及为什么需要部署自己的测试工具，它将带来什么益处、风险以及结果。

现在让我们来看看各种不同类型的测试工具，从测试管理工具开始介绍。

### 9.3.1 测试管理工具

测试管理工具用于管理测试。由于在测试过程中会生成大量信息，我们需要对这些信息进行管理，包括如下内容：

- 测试工件与测试依据的联系，例如需求、风险等。
- 在复杂环境中捕获的测试环境数据。
- 跟踪并行的测试执行，包括在不同地点的不同测试环境中执行的测试。
- 测试相关的度量：
  - 测试条件。
  - 测试用例。
  - 执行测试用例、测试套件、回归测试集以及其他测试过程相关度量元。
  - 测试用例、测试脚本、测试环境的数量。
  - 测试通过/失败的百分率。
  - 阻塞的测试用例数目（以及阻塞条件）。
  - 各种度量的趋势，如缺陷查找/修复的百分率。
  - 需求的数目和状态。
  - 测试脚本之间、测试设计之间的关系以及可追溯性。

测试管理工具自身也体现了某些组织概念，例如：

- 作为测试用例以及其他测试工作产品的中央资料库和驱动。
- 组织测试条件和测试环境。
- 存储测试用例并对其进行分组，构成回归套件和测试会话。
- 处理日志信息和失效信息。
- 命令环境重启和重新初始化。

测试经理、测试分析师和技术测试分析师会用到测试管理工具。

#### ISTQB 术语

**测试管理工具：**对测试过程中的测试管理和控制部分提供支持的工具。它通常有如下功能：测试件的管理、测试计划的制定、结果记录、过程跟踪、事件管理和测试报告。

### 9.3.2 测试执行工具

正确使用测试执行工具能降低成本、增加覆盖率以及增加测试的可重复性。由于回归测试工作量大且单调乏味，因此我们可以将回归测试自动化。

大多数测试执行工具通过执行脚本来进行测试，脚本通常是工具自定义的一种编程语言。工具通常都能精确地驱动按键或鼠标，并能检查图形用户界面或其他接口。这会导致测试脚本和预期结果变得很脆弱。

脚本是通过捕捉回放工具录制的，并以类似真实应用程序的形式构建。你可以使用



捕捉回放工具记录你的探索路径或其他非文档化测试，不过在这种情况下，测试脚本和预期结果很难维护。

测试执行工具使用一个比较器对预期结果和实际结果进行比较，预期结果来自于以前测试执行中捕捉的结果。通常要对比较器进行设置，避免比较那些变动比较大的域，例如日期和时间。

测试自动化的脚本与实际程序的构建方式类似，是以函数库或功能库的形式组织的，并由关键字表或数据表驱动。其中关键字方法也叫做关键字驱动的测试自动化。数据与脚本的分离能克服捕捉回放造成的维护问题。

如果编程能力和自动化架构的设计很糟糕，那么可能会导致测试自动化失效。除此之外，对脚本也需要管理，因为脚本本身也是程序，所以我们还要对其进行测试。

测试分析师和技术测试分析师会用到测试执行工具。

#### ISTQB 术语

**关键字驱动测试：**一种脚本编写技术，所使用的数据文件不单包含测试数据和预期结果，还包含与被测程序相关的关键词。用于测试的控制脚本通过调用特别的辅助脚本来解释这些关键词。

### 9.3.3 关键字驱动的自动化测试执行

先来解释一下什么是关键字驱动的测试。关键字代表了与系统进行的某些业务交互。例如，我们可能有一个关键字叫做“取消订单”，这个关键字代表了取消订单这个工作流程中所要执行的所有操作。我们将关键字组合在一起生成测试规程，等同于一个测试用例或一个完整的场景。

在自动化框架中，无论用的是什么工具，关键字最终都表示为一段可执行的测试脚本。脚本是模块化的。实现这样一个自动化框架需要测试人员具备熟练的编程技能。虽然为了构建这样一个框架需要预先付出很多，但这是值得的，这种方法能给我们带来很多益处。

首先，领域专家可以根据他们对系统的理解来定义关键字。因此，测试用例是基于系统用户的角度创建的，而不是从工具的角度或自动化人员的角度创建的。

一旦这样的自动化框架完成之后，领域专家就可以使用他们自定义的关键字和数据创建测试用例。每个人都可以运行这些测试用例。除非遇到测试失效，否则执行测试过程中不需要测试专业知识。领域专家可能需要对失效进行分析以确认问题出在被测系统身上还是测试框架有问题。

模块化的脚本和测试用例比线性的形式更加易于维护。如果一个程序界面发生了变更，那么一般情况下只需要对一个重用的脚本进行更新即可。

测试规格说明是独立于其实现的，因此可以换一个框架来实现它们。我们曾遇到过这样的情况，我们使用了3个框架在十几个操作系统上对一个查询工具的不同版本进行了测试。在编写框架的时候我们确实花了一点时间研究如何实现这些测试，但是一旦我

们搞清楚之后，我们就能很快地用不同的框架来实现这些测试。

关键字测试开发的大部分工作是由领域专家和测试分析师完成的，技术测试分析师参与框架的编写。

### 9.3.4 测试执行目标的案例分析

让我们来看一个实例，如图 9-3 所示。我们为一个客户实现了单元测试所需的测试执行工具。在项目的初期，我们为 4 类利益相关者识别了其相应的目标。管理层的主要目标如下：

- 降低回归次数，包括系统测试过程中和发布后。
- 增强发布时对变更的信心。
- 按时发布合格的产品。
- 增加测试和开发的效率。
- 降低由于变更引起的回归和其他副作用导致的延迟以及费用，尤其是降低系统测试期间的回归问题。
- 加快项目进度，尤其是降低系统测试期间的缺陷数目。

管理	开发团队
<ul style="list-style-type: none"><li>● 减少回归次数</li><li>● 增加对更改的信心</li><li>● 按时发布合格的产品</li><li>● 增加测试和开发的有效性</li><li>● 降低回归和变更副作用带来的延迟和费用</li><li>● 加快项目进度</li></ul>	<ul style="list-style-type: none"><li>● 对开发人员个体来说，通过执行自动化的回归测试来减少回归</li><li>● 实现单元测试的统一标准</li><li>● 减少回归测试对实验室资源的需求</li></ul>
RBCS	测试团队
<ul style="list-style-type: none"><li>● 使客户的单元测试项目成功</li></ul>	<ul style="list-style-type: none"><li>● 增加稳定的测试发布百分比</li><li>● 减少回归</li><li>● 提高测试效率</li></ul>

图 9-3 单元测试工具的目标

作为单元测试工具的主要用户，开发团队的目标如下：

- 对于开发人员来说，通过自动化的回归测试能降低回归的出现。
- 制定单元测试的统一标准，确保每个人对这项工作“完成”状态的理解是一致的。
- 降低回归测试所需的实验室资源，因为回归测试需要很多实验室资源以及开发人员的支持。

最后，处于开发团队下游的测试团队的目标如下：

- 增加稳定的测试发布的比率，因为糟糕的测试发布会消耗大量的测试时间。
- 降低回归缺陷。
- 增加测试人员的效率。

最后，我们的目标是帮助客户的单元测试项目获得成功。希望这个项目能作为一个成功的案例进行推广。当然，我们还想赚钱！

无论是什么类型的测试自动化项目，最好都能有类似于上述这样明确定义的目标。下一步是要为每个目标定义一些度量元（通常是指过程改进时的关键过程标识符）。



### ISTQB 术语

**测试执行工具：**用来执行实际测试的方法，包括手工的和自动的。

除了这些目标外，对于这个项目我们还需要注意一些重要的方面或限制。我们需要轻量级的、便于使用的工具以及过程。与程序员当前的工作量相比，创建和执行测试应该是一项次要的、增量的任务。这里无需任何重量级的过程。

我们需要一个能与当前工具和环境集成的工具。如开发环境，Visual C++；开发人员的 Windows 桌面应用程序；以及 TestDirector 数据库。

最后，根据这个项目的特性，我们要为单元测试级别的回归测试提供支持。这意味着我们得随时为多个回归测试以及开发人员提供支持，这些执行回归测试的版本是开发人员“私有的”，不过这些测试的定制内容都是类似的。

### ISTQB 术语

**调试工具：**程序员用来复现软件失败、研究程序状态并查找相应缺陷的工具。调试器可以让程序员单步执行程序、在任何程序语句中终止程序和设置、检查程序变量。

## 9.3.5 调试和排错工具

调试和排错工具能帮助我们定位缺陷。在某些情况下，缺陷位于用户界面上，因此能很容易地找到，不过在很多情况下缺陷与其症状的联系不是那么一目了然。调试工具包括日志、跟踪文件以及模拟环境。我们所说的调试工具实际上包括了调试工具和跟踪工具。

调试器允许程序员单步执行程序，检查意料之外的控制流或数据流。如果程序员怀疑缺陷位于某个点上或者想要检查某些变量，那么他可以在任何程序语句处暂停。调试器可以设置标志位并检查程序变量。

调试与测试有关，但并不是测试。类似的，调试工具与测试相关，但并不是测试工具。

技术测试分析师会用到调试和排错工具。

## 9.3.6 故障散播和故障注入

故障散播和故障注入是不同的技术，但是它们之间又有一些联系。故障散播工具与编译器有点像，它将缺陷散播到程序中去。这项技术用于检查测试发现类似缺陷的能力。当然，缺陷散播后的程序并不作为正式的代码。这项技术有时候也被称为变异测试 (mutation testing)。<sup>2</sup>

---

<sup>2</sup> 我没有用过这项技术，不过有人用过。Voas 和 McGraw 的“Software Fault Injection”一书中详细地描述了这项技术，因此如果对该项技术感兴趣，可以从这里开始。

**ISTQB 术语**

**故障散播工具：**在组件或系统中散播故障的工具（比如，故意插入一个故障）。

**静态分析器：**执行静态分析的工具。

**静态分析：**分析软件工件（如：需求或代码），而不执行这些工作产品。

故障注入通常是通过一个接口向程序注入有问题的数据或事件。例如，RBCS 的咨询人员能使用一个工具向文件注入随机数据。

技术测试分析师会用到故障散播和故障注入。

### 9.3.7 静态分析工具

静态分析工具能将静态测试过程中的一些工作自动化。它们会警示代码、需求等工作产品中存在的问题。例如，一个代码分析工具能标记出代码中存在的结构问题。而一个拼写和语法检查工具能大大降低需求规格说明的阅读难度。

当我们使用这类工具的时候常遇到的一个问题是假阳性。这里所说的假阳性是指工具警告的问题实际上不会造成任何危害。代码中可能存在大量的假阳性问题，大概每 5~10 行代码就存在一个这样的问题。

有很多策略能解决这个问题，例如只对新代码或刚修改过的代码进行静态分析。幸运的是，工具的供应商已经意识到了这个问题并正在着手解决这个问题。

技术测试分析师会用到静态分析工具。

### 9.3.8 动态分析工具

**ISTQB 术语**

**动态分析工具：**为程序代码提供实时信息的工具。通常用于识别未定义的指针，检测指针算法和内存地址分配、使用及释放的情况以及对内存泄露进行标记。

动态分析工具提供软件的运行时信息。它们能准确找出那些在静态分析中难以找到的或者在动态测试中难以定位的问题。这类工具能对指针的使用进行评估，不过最常见的用途是检查内存泄漏。在 C 和 C++ 语言编程中最有可能遇到内存泄漏问题，因为这两个语言要求程序员直接管理内存，而程序员常常会忘了这件事。

技术测试分析师会用到动态分析工具。

### 9.3.9 性能测试工具

**ISTQB 术语**

**性能测试工具：**一种支持性能测试的工具，通常有两个功能：负载生成（load



generation) 和测试事务 (test transaction) 测量。负载生成可以模拟多用户或者大量输入数据。执行时, 对选定的事务的响应时间进行测量并被记录。性能测试工具通常会生成基于测试日志的报告以及负载—响应时间图表。

性能测试工具通常包含两个主要部分。一个是负载生成器, 另一个是测量和分析组件。

负载生成器执行一个脚本, 该脚本模拟了用户的操作。有些脚本是通过捕捉回放工具录制的, 不过根据我的经验, 大多数脚本是手工创建的。脚本应该能够提供被测系统所需的任何数据。

大多数性能测试工具能同时模拟多个脚本, 因此能针对同一个系统并行模拟多种类型的用户。工具一般不会通过用户界面与被测系统交互, 而是使用 HTTP 或 HTTPS 等通信协议进行交互。这个时候, 测量组件会收集如下度量元:

- 模拟用户数。
- 模拟用户生成的事务数和事务类型。
- 特定事务请求的响应时间。

根据这些度量元, 可以生成各种报告, 包括负载响应时间图。

性能测试是很复杂的, 有众多重要因素需要考虑:

- 首先, 负载生成器所在的主机上是否有足够的硬件和网络带宽用于生成负载? 我就曾遇到过这样的情况, 在被测系统达到最大负载之前, 负载生成器的负载就已经最大了, 因此也就无法继续测试了。
- 第二, 工具的通信协议与被测系统的通信协议是否兼容? 这个工具能模拟需要模拟的任何操作吗?
- 第三, 这个工具是否能够创建并运行不同的用户操作?
- 最后, 这个工具是否具备所需的监视工具、分析工具和报告工具?

很多组织自行编写负载生成器用于可靠性测试, 也有一些组织会购买商业的性能测试工具或使用开源的工具。如果决定自己编写一个性能测试工具, 那么最危险的部分, 同时也是工作量最大的部分是测量和分析。

这里, 我还要提醒你的是, 大多数性能相关的缺陷是设计问题。我遇到过项目晚期才发现性能问题并最终导致项目失败的情况。因此, 如果性能是一项关键的质量风险, 那么一定要进行建模, 并在单元测试阶段就对关键的组件执行性能测试, 而不能等到系统测试阶段再这么做。

技术测试分析师会使用性能测试工具。

### 9.3.10 网站工具

网站工具是另一种常见的测试工具。

这类工具常被用于扫描网站中的无效链接。有些工具还能提供网站的链接树、下载的大小和速度、点击数以及其他一些度量元。有些网站工具能根据标准对 HTML 进行静

态分析。

测试分析师和技术测试分析师会用到网站工具。

#### ISTQB 术语

超链接测试工具：未在 ISTQB 术语表中定义。

### 9.3.11 模拟器和仿真器

如果你们曾看过宇宙飞行题材的电影，那么应该能想象得到这类电影其实是在模拟的环境中拍摄的。类似的，很多软件也是在模拟的环境中进行测试的。而模拟器则为我们提供了一个人工的模拟环境以用于测试。那么，为什么我们需要在模拟的环境中进行测试呢？可能的原因有系统的某些代码或某些部分还未完成、在真实环境中运行测试的成本太高、在真实系统中测试不安全。例如，飞机、宇宙飞船和核能控制软件在部署前都需要在模拟器中进行测试。你可以认为部署实际上是在真实环境中的第一次测试。

有些模拟器很复杂，它们能够注入故障、生成可重复的伪随机数据流等。对于模拟器测试，我的经验是无论用的模拟器有多好，当将软件移植到硬件中去的时候总会遇到各种各样的问题。因为很难模拟一些时间、资源限制和依赖相关的问题。

仿真器是模拟器的一种，它用软件来模拟硬件。我第一次遇到仿真器是在汇编语言课上，那时我正在加州大学洛杉矶分校的计算机科学学院读一年级。老师不允许我们直接在多用户的 VAX 机器上运行汇编程序，这的确是一个很聪明的决定，因为我们的程序肯定能够一遍又一遍地让这些机器崩溃掉。他们提供了一个仿真器，它就像是一个专用的汇编测试环境。

仿真器的一个潜在优点是它能为我们的测试提供支持（至少在仿真器包含这个功能的情况下是这样的）。我们学校提供的仿真器并没有提供什么很高级的测试支持功能，因为它的主要功能就是防止我们直接访问 VAX 硬件。而一个设计巧妙的、支持测试的仿真器能提供真实系统无法提供的功能，如跟踪、调试、记录以及其他监控程序运行的活动。

根据所需的仿真类型，测试分析师和技术测试分析师会用到这类工具。

#### ISTQB 术语

仿真器：一个接收同样输入并产生同样输出的设备、计算机程序或系统。

模拟器：测试时所使用的设备、计算机程序或者系统，当提供一套控制的输入集时它们的行为或运行与给定的系统相似。

### 9.3.12 自定义工具开发的案例分析

图 9-4 中显示的是我们为 IVR 综合系统项目设计的一个整体的系统集成测试套件的架构示意图。建议你在继续阅读下文之前先花点时间研究一下图 9-4。我们使用一个商业工具 SilkTest，与客户服务应用程序主机 PC 上的 Windows 图形用户界面交互。不过，我



们还有两个自定义的工具。

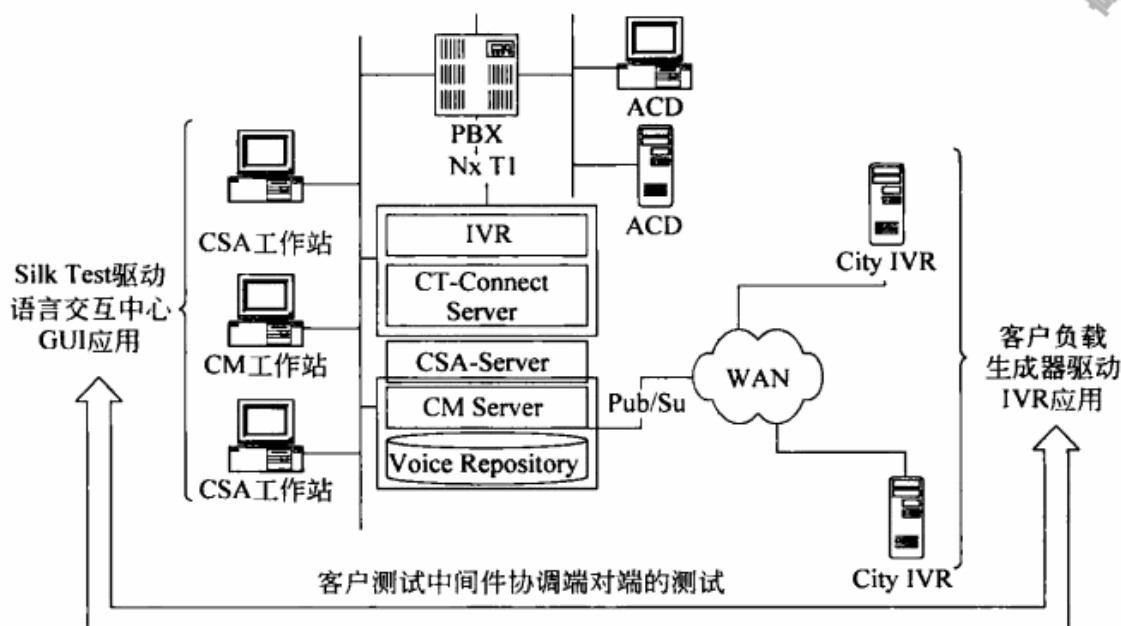


图 9-4 IVR 测试工具

一个是IVR接口测试工具。它可以通过电话线路向IVR系统的接口发送电话信号(人们称为双音多频信号)，模拟用户操作。在指定语音消息的格式之后，这个工具还能识别出收到的是否是正确的消息。我们得从头开始编写这样的一个工具，因此在这个过程中使用了一些开源组件，例如脚本语言TCL。当时并不存在符合我们要求的商业工具(至少在我们的预算范围内)。

另一个自定义的工具是用于协调IVR接口测试工具和GUI测试工具的中间件。我们称它为“伪通道”。它允许测试系统的两个部分互相发送消息。这样，他们就能对测试进行协调，包括完整的功能测试和性能测试。

## 9.4 认证考试模拟题

在每个章节结束之前，您可以尝试回答以下问题，加强对所学内容的理解，并准备ISTQB高级测试经理认证考试。

- 下面哪一项是与测试管理工具相关的风险？
  - 未经过培训的员工错误地使用了复杂的分类域
  - 对不恰当的测试进行自动化后产生了假阳性的结果
  - 测试经理深入了解了多个项目的发展趋势
  - 当程序员根据编码规范修改代码时引入了新的缺陷
- 在一份测试工具策略文档中你可以描述下列哪些主题？
  - 投资回报
  - 为特定工具选择试点项目
  - 从其他组织使用类似工具的经验中收集的好的方法
  - 开发被测系统的程序员列表

- E 每一个被测系统的质量风险项
3. 下面哪一个项目最有可能要求自定义一个自动化测试执行工具?
- A Windows PC 平台上的会计应用程序
  - B 在浏览器中运行的电子表格程序
  - C Linux PC 平台上的电子邮件应用程序
  - D iPod 上运行的一个嵌入式程序



## 个人技能和团队构成

高级大纲的第 10 章是个人技能和测试团队构成。该章节先是讲到了测试人员的个人技能，然后是内部的和外部的动态测试团队，最后讨论了如何激励测试人员和测试团队，以及如何就测试进行沟通。本章共包括 6 个部分：

1. 概述。
2. 个人技能。
3. 动态测试团队。
4. 组织的测试选择。
5. 激励。
6. 沟通。

下面一一讲解这些小节，看看它们与测试管理有什么关联。

### 10.1 概 述

#### 学习目标:

能回想起该部分内容即可。

本章主要讨论与测试管理相关的主题。因此，它主要是从测试经理的角度出发进行阐述的。作为测试经理，我们需要了解的主题有：

- 测试人员的个人技能。
- 动态测试团队。
- 测试团队的组织。
- 测试团队激励。
- 沟通问题。

在测试团队中，所有测试人员都应该小心地处理他们之间的关系，以及与项目团队中其他成员的关系。

## 10.2 个人技能

### 学习目标:

(K3) 使用给定的问卷从多个角度对团队成员的优缺点进行分析: 软件系统的使用、专业领域和商业知识、系统开发范围、软件测试和人际交往能力。

作为测试经理,我们必须确保测试团队的技能组成是恰当的。这好像是显而易见的,实则不然,你总是会惊奇地发现你的团队中有很多完全不符合要求的人员。

事实上,在写本章节的时候,我们正在应付一个对书感到不满意的客户,这个客户也是一个测试人员,由于在书籍中没有找到他想要的模板,因此他希望退回那些书。其实那些书中是包含了测试计划、测试用例和测试规程模板的,只不过这些模板的使用对于这位客户来说显然是困难了点,他不想进行太多的思考也不具备相应的技能。他在信中写道:“我要退还这两本书。我原本指望在书中找到测试计划、测试用例、测试规程的模板,这样我就能像‘填空题’一样往模板中填写我们公司的测试项,就可以得到一份测试计划或测试用例或测试规程。但这些书没有提供我想要的东西。我是一个防御设备的供应商……”

想想吧,士兵、水手或者海军陆战队员的安全取决于军用设备能正常工作,而这些设备大多数都包含了软件。这位老兄甚至都不愿意花点时间去看看如何使用我们提供的详细模板。哦,不,这里要考虑的东西太多了。

作为测试经理,我们有多种方法能确保测试团队的技能组成是恰当的。一种是设法确保团队内技能的平均分布,也称为通才团队方法。或者我们也可以允许团队成员发展他们的专长,这样整个团队在所有的领域都非常强,这种方法也叫做专家团队。你可以想象得到存在无数种这样的方法。无论你使用哪种方法,一个叫做技能清单的方法都会对你有帮助。

随着经验的积累以及参加培训,测试人员的技能水平会有所提高。仅仅依靠经验积累获得的技能水平的提高,无论对个人还是对团队而言,都是不可靠的。而通过参加培训提高技能水平的方法效率比较低,而且很容易导致一个团队有很强的理论背景,但处理实际测试问题的能力很低。最理想的方法就是先进行目的明确的培训,然后马上在实际项目中对培训中学到的知识进行实践。

一个测试人员或测试团队应该具备什么样的技能呢?应该从下面 3 个主要方面考虑。

- 软件系统的使用,包括领域知识或业务知识。换句话说,测试人员是否理解系统应该做些什么?
- 了解软件开发,包括分析、开发、技术支持和技术文档。换句话说,测试人员是否了解系统编写、系统部署以及为系统提供支持相关的问题?
- 了解软件测试。换句话说,测试人员是否能用最佳的方法解决他们要面对的各种问题?



在接下来的3小节中会一一进行解释。

### 10.2.1 测试技能

所有测试人员，无论他们在团队中是什么角色，都需要优秀的人际交流的能力。他们应乐于提出和接受批评。由于测试工作的压力以及经常需要面对一些坏消息，因此测试人员在情感上都应该有一定的承受能力，不至于在工作中表现得沮丧、玩世不恭或麻木不仁。

测试人员，尤其是测试经理，也是项目团队中的一个角色。为了完成工作并使你发现的问题得到适当的处理，通常你需要具备一定的影响力和谈判能力。

许多测试人员在技术上是非常有竞争力的，有些人甚至可以被称为技术天才，但他们会发现由于糟糕的人际交流能力而无法在项目中发挥所长，无法取得应有的成功。如果发现经常发生以下情况：自己经常被忽视、与其他项目成员发生激烈争执、被认为是敌意的、没人通知你重要的项目信息、在其他人的电子邮件黑名单中、一直未获得职位晋升，那么你应该仔细地审视自己的人际交流技能，诚恳地评估你所扮演的角色。

测试基本上就是在生成信息，生成大量的信息。测试工作包括生成、编辑、评审、阅读以及分析一系列的技术文档和度量。因此，测试人员需要具备优秀的组织能力；尤其是那些混乱、无组织的项目。

测试人员必须了解项目的各个方面，但同时又专注于重要的细节。我听到过很多“粗线条”的例子，忽视细节导致了他们的失败。我也知道有些人属于“细节导向”的类型，但由于关注了一些不重要的事情而导致了他们的失败。因此不要老是对一些无关紧要的事情吹毛求疵。这种了解全局、关注细节的能力对于测试经理尤其重要。

由于测试生成大量信息，因此您的表达能力决定了您作为一个测试人员的效率。要具备很强的书面和口头的表达能力。这些技能对测试经理尤其重要。除了基本的个人技能外，测试分析师和技术测试分析师还必须具备一些特定的测试技能：

- 阅读和分析系统的需求规格说明、设计规格说明、用例和其他相关文档。
- 参与产品风险分析和项目风险分析活动。事实上，我发现很多测试人员正在组织这样的活动。
- 运用基础级大纲和高级大纲中的那些概念，设计出有效的、高效的测试集。所需技能包括创建测试数据、选择测试准则、定义预期结果并在某些情况下参与测试自动化。
- 运行测试、评估结果、并有效地记录测试结果。所需技能包括记录测试状态、测试事件和测试影响事件。同时还需要测量一些度量元。

对于测试经理而言，应具备的特定测试技能如下：

- 对项目进行管理，因为测试通常是整个项目中的一个子项目。测试经理应该具备所有项目管理所需的技能。
- 组织各种与测试计划相关的活动。所需技能包括风险分析、测试评估和制定测试计划。同时还包括制定有效的测试策略以及参与组织测试方针的制定。
- 跟踪测试进度。所需技能包括测量度量元、记录日志、分析度量元、评估现实与

项目计划和测试计划的差别，以及建议并实施恰当的测试控制活动。

- 报告测试结果。所需技能包括创建、展示并解释第3章中提到的各种图表和报告。与项目的利益相关者有效地交流测试的结果是一项很关键的技能。如果因不具备这项技能而导致项目团队拒绝或误解你的发现，那么你在测试计划、测试设计，以及测试执行阶段所做的一切都白费了。

对于那些想做测试经理的测试分析师和技术测试分析师，应该在获得这个职位之前就尽早培养上述技能。不然他们很有可能因无法适应角色的转变而导致失败。

### 10.2.2 技术和软件技能

有些测试人员认为我们不应该知道如何编程，也不应该有很强的技术技能。他们的理论是，当测试人员理解了缺陷是如何变成失效的机制之后，他们会对失效表现得更宽容。

的确，当测试人员长时间测试同一个系统，他们就会对失效更宽容。我将这种现象称为环境适应。即使是完全不懂技术的测试人员也会发生这种情况。作为一名测试经理，你的工作就是留意并预防这种现象。根据我的经验，我认为环境适应这种现象并不是由于技术知识造成。技术知识只是增强了我们的理解能力和效率，同时也使得我们能与程序人员更好地对失效和缺陷进行交流。<sup>1</sup>

了解需求分析、设计和编码的测试人员自然也了解缺陷的生命周期。换句话说，他们知道程序员是如何引入缺陷的，甚至能在第一时间预防缺陷。

了解技术支持的测试人员对于用户体验、用户和客户期望以及可用性问题会有比较深入的了解。事实上，有些客户的测试团队完全是由以前的技术支持人员组成的。如果技术支持人员同时具备了其他领域的技能，那么这种人事安排将是可行的。但如果他们不具备其他所需技能，他们会表现出技术支持人员的一些特性，例如他们会像救火一样试图立刻解决问题等，这将给测试团队造成很大的问题。

最后，我在第9章提到过，许多测试自动化任务要求具备软件开发经验。因此，应该确保你的团队成员有相应的知识和经验。

### 10.2.3 用户、业务以及领域技能

真正认真做测试的人需要能从用户的角度来思考问题，当然，这有可能使他们问一些没有意义的问题。他们知道如何使用系统，并理解失效的优先级，这有助于质量风险分析和事件管理。他们清楚系统的“正确的行为”应该是怎样的，或者至少清楚他们希望中的行为应该是怎样的，因此这些想法能作为测试准则的一部分。

业务和领域专家，无论他们是否是用户，都能增进我们对系统的理解。他们知道业务中各种重要的功能，以及哪些行为会对业务产生影响。因此他们能为质量风险分析和

---

<sup>1</sup> 并不是所有人都认同我的这一观点。RBCS的高级助理 Judy McKay 在评审本书的时候注释道：“这个观点很有意思，但是我的经验是，技术更强的人对缺陷的根源更加了解，自然能找到更多会惹恼用户的缺陷。同时我也喜欢有一些纯黑盒的测试人员执行测试，确保我的团队不会因为技术过强而忽视了从用户的角度考虑问题。”



事件管理提供帮助。

除了质量风险分析和事件管理，用户和领域专家还能参与其他测试活动。他们能帮助测试经理确定测试活动的优先级，尤其是当各种项目限制使得测试团队无法完全按照风险优先级执行测试的时候。他们还能提供或帮助创建一些有现实意义的测试数据和测试用例。他们还可以对用例进行验证，或更好的情况下，直接提供用例。

当然，作为一名测试经理，应该慎用用户和领域专家的这些能力。一方面，这些人通常还有其他工作，且参与测试活动会使他们分心甚至惹怒他们。我曾过于依赖领域专家，而这些专家是受另外一名经理管理的，这导致有人抱怨我的管理方式是“吸血管理”以及我的团队技能组成不合理。

另一方面，用户和领域专家有时候感到他们有权力对项目成员发号施令。这时候，必须小心地管理这些人的干涉行为，不然就会发现一些项目之外的人员和经理在质疑、微管理以及命令你的测试工作。作为一名测试经理，必须能够巧妙地请求帮助，而又不丧失对自己团队的控制。

#### 10.2.4 技能清单和管理

我们已经讲解了测试人员应该具备的个人技能。那么，作为一名测试经理，应该如何测量、管理和培养这些技能呢？我们将会通过一些图片，讲解如何做到这一点。这些图描述的是一个技能清单。这个清单基于我们假设的一个项目，开发一个基于浏览器的文本处理器。

在这个电子表格中，左侧是一个技能列表。图 10-1 中是一些通用的专业技巧。而后文中的 3 个图给出的是测试技能、领域技能和技术技能。再次说明，这个例子基于一个虚拟的项目。你需要根据任务分析，即你的团队需要做什么以及应该能做什么，然后自定义一个关键技能列表。

表格顶部的说明文字解释了表中一些项的含义。我们定义了 4 个等级，并对各种职位所需的技能，以及当前团队成员的技能进行了评定：

- 0 表示这个人对该项技能不了解或没有经验。
- 1 表示这个人对该项技能有一定的了解以及经验，不过在使用这项技能的时候需要咨询别人。
- 2 表示这个人对该项技能很了解并且经验丰富，在使用这项技能的时候也无需咨询他人。
- 3 表示这个人的这项技能是专家级的，并能指导别人如何使用这项技能。

表格中还有另两种标记类型：

- R 表示该职位需要这项技能（通常还有等级要求）。D 表示期望有这项技能，但并非必须的。如果在招聘的过程中无法区分需要的和期望的技能将会造成很多困扰，因此你要记住这些区别。
- TM 表示测试经理、TA 表示测试分析师、TTA 表示技术测试分析师。这些标题的含义与高级大纲中术语的含义一样。

技能和资质	说明	0=无	1=了解一点	R=要求		
		2=了解	3=精通	D=期望		
基本要求		TM=测试经理	TA=测试分析员	TTA=技术测试分析员		
	TA 最低要求	TTA 最低要求	TM Irit Kesef	TA Charlotte Wright	TTA James Malone	团队 最低要求
教育						团队 平均要求
理学士(或更高)	D	D				
	D	D				
测试培训或认证						
工作经验						
测试角色	5R	5R				
	D	D				
非测试, 计算机	D	D				
	D	D				
非计算机, 领域						
非计算机, 非领域						
职业技能						
口头表达	2R	2R			0	#DIV/0!
	3R	3R			0	#DIV/0!
非正式书写	D	D			0	#DIV/0!
	D	D			0	#DIV/0!
正式书写	R	R				
	R	R				
继续教育	2R	2R			0	#DIV/0!
	2R	2R			0	#DIV/0!
测试团队建设、交叉培训	2R	2R			0	#DIV/0!
	2R	2R			0	#DIV/0!
跨功能关系构建	2R	2R			0	#DIV/0!
	2R	2R			0	#DIV/0!
阅读(记忆、理解、分析)	1R	1R			0	#DIV/0!
	1R	1R			0	#DIV/0!
业务、技术趋势						

图 10-1 技能清单和管理——基本要求

现在,我们再看看最左侧的技能列表。这个列表的右侧两列的标题是测试分析师\_最小等级和技术测试分析师\_最小等级。在这两列中,对于每一个关键技能,我们都给出了所需的最小技能等级。大多数技能的评级是上面提到的0~3级,而对于“经验年限”这项的评级自然是指年限。

在填写这些列的时候,可以一项一项地问自己,“该职位的这项技能应该达到什么等级?”你可能想要填入一个理想值。但这是不现实的。你应该填一个最小值,即如果某人这项不达标那么你就不会录用他。再次说明,要分清期望的和需要的技能的区别。

另外,在按照技能清单进行评定的时候,对自己的团队成员也要评定,看看他们是否满足最低标准。如果确实有很多人未满足标准,也不用着急;这并不是说不得不解雇一批人然后再招聘新人。而是表明需要采取措施来培养团队的一些关键技能。

在图中间的那三列的地方,应该对团队中的每一个成员进行评定,每人一列。记住使用我们刚才提到过的评级方式,公正地、诚实地进行评定。每个家长在评定自己的孩子的时候都会以各种方式夸大事实,而在这里不应该这么做。

您可能已经注意到,在这个例子中我将测试经理也包含在内。如果经常向其他人分享你的知识,那么我建议您将自己,即测试经理,也包含在表格内。换句话说,如果最近您也做过很多测试工作——作为一名真正的实践者,并且还是该领域的专家,那么即使您现在很少亲自动手,还是能花点时间去指导别人。然而,不要不懂装懂,也不要



兜售一些过时的信息，因为这会让你像个傻子，你在“培训”别人已经知道的东西或者不再必要的技能。

在了解了这些列之后，想必表格右侧的两列的含义你也已经明白。团队最低表示你的团队内至少有一个人是弱的。这个标识很有意思，如果你期望的团队是一个通才团队，那么这一项就表示有某些任务需要特定的人去完成，或至少有些工作那个弱的人无法完成。而如果团队是一个专家团队，那么这个标识应该改为团队最高，因为应该保证至少有一个人在这项技能上有专家级的水平（即等级3）。

团队平均也是一个关注平均能力的度量元，即你的团队的平均水平有多高。如果你的团队是通才团队，那么你期望的平均值也许会等级2。而如果是一个专家团队，那么你的度量应该是每个技能中等级为3的人数。换句话说，你要知道你的团队内有多少专家？

评估完成之后，列出团队的薄弱之处。然后，制定相应的计划，通过培训或分配任务来克服这些弱点。阶段性地对团队技能进行再评估，更新这个清单并改进计划。这是一项持续性的工作，而非一次性的。至少应该每6个月评估一次。

图10-2列出了测试所需的技能列表。您可能已经注意到这个表格中存在一些问题。

技能和资质	TA 最低要求	TTA 最低要求	TM Irit Kesef	TA Charlotte Wright	TTA James Malone	团队 最低要求	团队 平均要求
测试技能							
通用要求							
测试标准和成熟度	2R	2R				0	#DIV/0!
版本控制和配置管理	1R	1R				0	#DIV/0!
计划							
估算和质量费用	D	D				0	#DIV/0!
文档	D	D				0	#DIV/0!
质量风险分析和管理	D	D				0	#DIV/0!
设计与开发							
行为的(黑盒)	2R	2R				0	#DIV/0!
结构的(白盒)	D	1R				0	#DIV/0!
静态的(评审和分析)	D	2R				0	#DIV/0!
性能(建模/模拟/测试)	2R	D				0	#DIV/0!
测试自动化							
COTS执行工具	3R	D				0	#DIV/0!
COTS测试管理	D	D				0	#DIV/0!
测试数据生成器	1R	D				0	#DIV/0!
测试执行							
手动(根据文档或随意的)	D	3R				0	#DIV/0!
自动化	3R	D				0	#DIV/0!
测试状态报告和度量	2R	2R				0	#DIV/0!
平均测试技能			#DIV/0!	#DIV/0!	#DIV/0!	0.0	#DIV/0!

图10-2 技能清单和管理——测试技能

例如，是技术测试分析师而不是测试分析师需要测试自动化技能。在制作这样的工作表时很容易犯这种错误，不过这种错误可以自我修正，因为当您在评审会进行技能评估的时候会注意到这些问题。

图 10-3 中列出了测试所需的领域技能。同样的，在这个表格中还是会发现一些错误。例如，测试分析师的领域技能通常应该比技术测试分析师的要求高。在使用这个技能清单的时候您将会发现并修正这些问题。

技能和资质	TA 最低要求	TTA 最低要求	TM Irit Kesef	TA Charlotte Wright	TTA James Malone	团队 最低要求	团队 平均要求
领域知识							
文本处理							
Windows应用程序	1R	2R				0	#DIV/0!
Linux/UNIX应用程序	D	D				0	#DIV/0!
Macintosh 应用程序	D	D				0	#DIV/0!
图像、图形和表格	1R	2R				0	#DIV/0!
文档管理							
Windows 应用程序	D	D				0	#DIV/0!
Linux/UNIX应用程序	D	D				0	#DIV/0!
Macintosh 应用程序	D	D				0	#DIV/0!
文档交换							
Windows 应用程序	D	D				0	#DIV/0!
Linux/UNIX 应用程序	D	D				0	#DIV/0!
Macintosh 应用程序	D	D				0	#DIV/0!
打印							
颜色与带宽	D	D				0	#DIV/0!
激光、喷墨、其他	D	D				0	#DIV/0!
发布/装订	D	D				0	#DIV/0!
网站发布							
HTML	D	D				0	#DIV/0!
PDF	D	D				0	#DIV/0!
平均领域知识			#DIV/0!	#DIV/0!	#DIV/0!	0.0	#DIV/0!

图 10-3 技能清单和管理——领域技能

最后，图 10-4 列出了测试所需的技术能力。

### 10.2.5 个人技能练习

作为 HELLOCARMS 系统的测试经理，您需要对团队进行评估。你应该如何对先前的那些技能列表进行修改，以用于对这个银行项目团队的评估？列出应该保留的技能和应该修改的技能。

如果是在课堂上，那么 3~5 人组成一组讨论。等所有团队完成练习后，团队之间再进行交流。如果只是您一个人，那么就只能自己做练习了。



技能和资质	TA 最低要求	TTA 最低要求	TM Irit Kesef	TA Charlotte Wright	TTA James Malone	团队 最低要求	团队 平均要求
技术知识							
编程							
C/VB (3GL)	1R	D				0	#DIV/0!
Java/C++ (OO)	1R	D				0	#DIV/0!
Shell 脚本编程	2R	D				0	#DIV/0!
代码复杂度和度量	1R	D				0	#DIV/0!
操作系统							
Windows	1R	1R				0	#DIV/0!
Linux/UNIX	1R	1R				0	#DIV/0!
Mac OS	D	D				0	#DIV/0!
网络/互连技术							
TCP/IP、FTP、RCP (网络架构)	1R	1R				0	#DIV/0!
浏览器 (Firefox、IE等)	1R	1R				0	#DIV/0!
网络应用架构	1R	1R				0	#DIV/0!
网络硬件	1R	1R				0	#DIV/0!
系统和服务器							
网站/应用服务器	1R	1R				0	#DIV/0!
数据库服务器	1R	1R				0	#DIV/0!
平均测试能力			#DIV/0!	#DIV/0!	#DIV/0!	0.0	#DIV/0!

图 10-4 技能清单和管理——技术能力

建议用 30 分钟的时间来做这个练习，包括讨论时间。

10.2.6 个人技能练习参考答案

在表 10-1 中列出了测试所需的 4 种主要技能。

表 10-1 适用于 HELLOCARMS 项目的技能列表

基 本 要 求	
技能和资质	解释/注释
教育	
理学士（或更高）	
测试培训或认证	
工作经验（年限）	
测试角色	
非测试，计算机	
非计算机，领域	
非计算机，非领域	
专业技能	
口头表达	

续表

基 本 要 求	
技能和资质	解释/注释
正式书写	
非正式书写	
继续教育	
测试团队构建/交叉培训	
跨功能关系构建	
阅读（记忆、理解和分析）	
业务/技术趋势	
测 试 技 能	
技 能	解释/注释
通用	
测试标准和成熟度	
版本控制和配置管理	
计划	
估算和质量成本	
文档	
质量风险分析和管理	
设计和开发	
行为的（黑盒）	
结构的（白盒）	
静态的（评审和分析）	
性能（建模/模拟/测试）	
测试自动化	
COTS 执行工具	
COTS 测试管理	
测试数据生成器	
执行	
手动（根据文档或随意的）	
自动化	
测试状态报告和度量	
领 域 知 识	
技 能	解释/注释
文本处理	用下面的内容替代
Windows 应用程序	
Linux/UNIX 应用程序	
Macintosh 应用程序	



续表

领 域 知 识	
技 能	解 释/注 释
图像、图形和表格	
银行	
房屋抵押贷款	
反向抵押贷款	
信用限额	
抵押贷款	
支行的一般事务	
文档管理	
Windows 应用程序	用下面的内容替代
Linux/UNIX 应用程序	
Macintosh 应用程序	
信贷	
决策系统	
信贷部门	
文档交换	
Windows 应用程序	用下面的内容替代
Linux/UNIX 应用程序	
Macintosh 应用程序	
规范	
沙滨法案	
数据隐私	
州法	
打印	删除这部分
颜色与带宽	
激光、喷墨、其他	
发布/装订	
网站发布	删除这部分
HTML	
PDF	
技 术 知 识	
技 能	解 释/注 释
编程	
C/VB (3GL)	根据 HELLOCARMS 系统的实现情况选择其中之一
Java/C++ (OO)	
Shell 脚本编程	

续表

技术知识	
技 能	解释/注释
代码复杂度和度量元	
操作系统	
Windows	
Linux/UNIX	
Mae-OS	并没有用到
网络/互连技术	
TCP/IP、FTP、RCP（网络架构）	这些技能的选择取决于格罗班克银行的标准和架构
浏览器（Firefox、IE 等）	
网络应用架构	
网络硬件	
系统和服务器	
网站/应用服务器	
数据库服务器	

我认为应该删除的技能用删除线标识（如技能），应该增加的技能用加粗标识（如技能）。既没有用删除线标识，也没有加粗的那些技能是保留项。在有些地方我增加了一些注释，方便理解。

你可以看到，除了领域技能那个部分，其他部分很少有改动。这是因为测试不同应用程序所需的测试技能比较类似，且所有基于浏览器的应用程序，无论是文本处理还是银行系统，它们的基本技术都是一样的。

## 10.3 动态测试团队

### 学习目标:

（K3）进行对比分析以确定组织内某个空缺职位所需要的技术和软实力。

作为一个测试经理，测试团队的发展也许是您所管理的最关键的测试过程了。如果测试团队得到良好的发展，那么当您犯了一些测试经理常犯的典型错误，团队能掩护你。例如，如果您没有邀请关键的项目利益相关者来改进你们团队的缺陷报告的质量（这是测试经理犯的典型错误），团队成员就会提醒您注意这个细节。

相对的，如果您没有很好地管理和发展测试团队，那么即使您把其他事情做得很好，团队还是会遇到一些严重的问题。雇佣一个你不应该雇佣的人会大大降低测试团队的效能。

一般新的测试人员都是通过雇佣而来的，不过有时候也可能是从组织的其他部门调过来的。然而，对于团队以这种形式成长我都持怀疑态度。因为如果有人从另一个团队



被调到您这个团队，那么这其中必然有一个原因。在理想情况下，可能是这个人的技能更适合您的团队，这样当然所有人都能受益。不过根据我的经验，更多情况下是因为其他经理想要摆脱这个问题员工，但又不想面对繁琐的程序去解雇他。您应该像雇佣新员工一样，对内部调度而来的员工进行严格的评估。

一个恰当的团队成员不仅仅是个人技能问题。还必须从团队的角度考虑问题。您可以使用我们上面提到的技能清单，挑选一个拥有互补技能的员工来填补团队的弱点并指导他人。

然而，您必须考虑一个人的性格。例如，您是否希望有一个人能锲而不舍地弄清需求、大的缺陷等问题。如果需要这样一个人，那么就不应该雇佣那种性格内向、害羞的人。

一个好的测试团队内应该包含了各种类型的性格。这就像我刚才提到过的，一个团队内应该有各种特殊技能的成员一样，这些人能够互补。

作为一名测试经理，应该将测试团队的发展过程视为一个持续的过程。这个团队应该适合各种项目。另外，他们应该能与项目团队里面的其他成员良好地沟通。

当向团队内添加人员时，无论是新雇用的还是从组织内转移过来的，应该确保从一开始这种体验都是正面的。我所说的“正面的”体验包括了新来的团队成员、测试经理，以及团队的其他成员。

这需要您这个测试经理与他有一段时间的密切交流，让他知道要做什么，并得到适当的监管。事实上，很多经理喜欢在不告诉新手该干什么的情况下直接把他们“放在火上烤”，不要犯这类错误。

您应该明确定义新员工的角色。分配的任务应该是他们力所能及的，而不是会导致他们失败的任务。有时候，尤其是在非常动态的工作环境中，您会为您的团队成员布置一些“挑战性目标”，不过我建议新员工新来的90天内，您安排的任务应该是在他们的能力范围之内的。可以根据技能清单来决定应该安排什么任务。

在精明、强硬的管理思想大行其道的今天，有些人会认为我这种做法太娇惯、温和了。我承认这种说法。我认为个人的成功是与团队的成功联系在一起的。如果许多人都遭遇失败，这对整个团队而言没什么益处，甚至相反会有不良的影响。

在过去的10年间，我发现有一种趋势，即管理越来越麻木不仁，人们喜欢用基于恐惧的管理模式。我曾听到有些经理，从运营官到基层经理，表示个人是可以拓展的并且如果他们不喜欢某个人的工作，他们就会设法摆脱这个人。有时候就让人觉得这些人完全不考虑他们的员工是有家人需要喂养的、是有个人抱负的，以及是需要过生活的。不过这些经理慢慢发现他们团队的成员没有什么忠诚可言，工作也不会尽全力。

在RBCS，我很幸运能与我们的员工和助理们一起工作。我们的测试团队能，并且也确实这么做的，与客户和经理们一起共渡难关。我将这归功于一种支持性的管理模式。

因此，我认为作为一名测试经理，想方设法成为一个好的领导也是工作之一。除了

要为员工提供支持之外，还要考虑他们合适的职业发展。个人的成功源自于能将你分配的角色和任务与他们的技能和个性恰当地结合在一起。

不能只关注团队的技能，而得做一些实际的工作。确保所有的个人技能的组合能满足团队任务的需要。第3章中讨论过的测试方针应该定义了测试团队的任务。如果您没有这样一个管理层认可的方针，无论是文本的还是口头的，将无法培养测试团队，因为没有明确的定义，好的团队应该是什么样的或者好的团队应该做什么。

有时候，RBCS 也会帮客户进行员工培训以及寻找合适的员工，我们发现人们总是试图找一些完美的员工。10年的Java编程经验、15年的测试自动化经验、5年的领域经验，然后愿意在一个基础职位上每周65小时地工作。

问题出在我先前提到过的，没能分清楚工作所要求的和期望的技能的区别。有些人会说，“为什么不把合格线设得高一点，这样就能吓退那些不合格的人，而合格的人会觉得他们能达到这一要求呢？”

错。实际情况并不是像您想象的那样。很多满足要求的人员会严肃地思考，并选择不尝试这份工作，这样您就没得选择了。而很多不满足这些不切实际的要求的人会想，“这很愚蠢，没有人能符合这些要求，既然我和别人都不符合，那为什么不试着申请一下？”然后你就会花费大量的时间去筛选简历。

因此，应该设置切合实际的要求，并按照这个要求进行招聘。注意，我并不是说可以招收那些不具备竞争力的，或甚至更糟的问题制造者。我的意思是应该对您的团队成员进行培养，在理想情况下，新员工会有这些能互补的技能。

当然，完美的员工肯定是存在的。我和我的合作伙伴曾为RBCS招聘到一个完全符合我们期望的员工。不过，发生这种情况的可能性小得可怜。

您要找的团队成员应该是那些符合要求的，而不是完美的成员。不过，久而久之，您可以完善你的团队。这包括采用好的招聘决策；构建一个强大的、平衡的团队。

每雇佣一个员工，都应该这么问自己：新雇佣的员工能带来什么新的技能、经验、态度？这个员工是否能对其他成员进行培训，将他的技能和经验传递给别人？这个人有什么弱点？你要怎么处理这些弱点？

第8章我们讨论了持续的测试过程改进。现在，我们讨论的是持续的测试团队改进，这两者是同等重要的，并在有些情况下，后者可能更重要。

后面有4个图，这里，我们继续使用上节中的例子，不过这次是通过使用技能清单对一个假设的测试团队进行缺口分析。这个团队只有3个人：Irit、Charlotte和James。不过，这已经足够用于说明这个技术了。

工作表的左侧是完成重要的测试任务所需的技能。这些技能主要分为4个小组：基本要求、测试技能、领域技能和技术能力。我们将用等级0~3级对每个成员进行评定，找出需要改进的技能。其中等级2表示这个人对该项技术很了解，使用的时候不需要咨询别人。

假设我们希望构建一个通才团队。在这种情况下，我们希望对两种特殊情况进行评估：



- 有一个或多个个人的技能评分小于 2。如果存在这样的问题，表格右侧的“最小值”列会揭示出来。
- 团队的整体技能评分只有或略高于 2。如果存在这样的问题，表格右侧的“平均值”列会揭示出来。

现在，让我们对图 10-5 中的工作表进行评估。

说明	0=无	1=了解一点	R=要求			
	2=了解	3=精通	D=期望			
	TM=测试经理	TA=测试分析员	TTA=技术测试分析员			
技能和资质	TA 最低要求	TTA 最低要求	TM Irit Kesef	TA Charlotte Wright	TTA James Malone	团队 最低要求
基本要求						团队 平均要求
教育						
理学士(或更高)	D	D	BS/MBA	BA	BS	
测试培训或认证	D	D	ISTQB	ISTQB	ISTQB	
工作经验						
测试角色	5R	5R	12	10	7	
非测试，计算机	D	D	2	3	4	
非计算机，领域	D	D	0	2	0	
非计算机，非领域			0	0	0	
职业技能						
口头表达	2R	2R	3	2	2	2
非正式书写	3R	3R	3	3	3	3
正式书写	D	D	3	2	2	2
继续教育	R	R	Y	Y	Y	
测试团队建设、交叉培训	2R	2R	2	2	2	2
跨功能关系构建	2R	2R	3	3	2	2
阅读(记忆、理解、分析)	2R	2R	2	2	2	2
业务、技术趋势	1R	1R	3	1	1	1

图 10-5 使用技能清单进行缺口分析——基本要求

在图 10-5 中，测试分析师、Charlotte 和 James 在业务和技术趋势这一项中比较弱。增加一些时间对这些趋势进行研究对工作应该能有所帮助。或许，Irit，作为一名测试经理，可以安排这两个人对工作相关的业务和技术的趋势进行研究和报告？其他一些平均值为 2 的技能看起来问题不大，因为其中每个人的评分都是 2。接下去的 3 个图是对测试技能、领域技能和技术能力的评估。

那么，通过技能清单和缺口分析我们可以做些什么？通过这些工作表，可以直接得出招聘所需的工作描述。可以列出工作所需的技能，尤其是当前团队比较薄弱的技能。

另外,还可以根据这个评估结果制定一个培训或交叉培训计划,从那些最薄弱的技能开始。

让我们看看测试技能的缺口。在图 10-6 中,这个团队的测试技能与我们的期望相比还是比较薄弱的。在安排 Charlotte 和 James 对趋势进行研究之前,Irit 应该对这个工作表中的技能按重要性进行排序,制定计划在接下去的 6 个月内系统地、全面地提升这些技能评分。

技能和资质	TA 最低要求	TTA 最低要求	TM Irit Kesef	TA Charlotte Wright	TTA James Malone	团队 最低要求	团队 平均要求
测试技能							
通用要求							
测试标准和成熟度	2R	2R	3	1	1	1	1.7
版本控制和配置管理	1R	1R	2	1	1	1	1.3
计划							
估算和质量费用	D	D	2	1	1	1	1.3
文档	D	D	2	1	1	1	1.3
质量风险分析/管理	D	D	2	1	1	1	1.3
设计与开发							
行为的(黑盒)	2R	2R	1	3	1	1	1.7
结构的(白盒)	D	1R	1	0	2	0	1.0
静态的(评审和分析)	D	2R	1	3	2	1	2.0
性能(建模/模拟/测试)	2R	D	1	2	2	1	1.7
测试自动化							
COTS执行工具	3R	D	1	1	3	1	1.7
COTS测试管理	D	D	1	3	2	1	2.0
测试数据生成器	1R	D	1	2	1	1	1.3
测试执行							
手动(根据文档或随意的)	D	3R	3	2	2	2	2.3
自动化	3R	D	1	1	3	1	1.7
测试状态报告和度量	2R	2R	3	3	1	1	2.0
平均测试技能			1.6	1.6	1.6	1.0	1.6

图 10-6 使用技能清单进行缺口分析——测试技能

现在,让我们看看领域技能中存在的缺口。在图 10-7 中,一眼看下来的第一感觉肯定是太糟糕了。不过,注意其中很多技能标识的是“期望的”而不是“需要的”。因此,对于领域技能来说,Irit 应该制定计划,争取在下一年中解决那些严重不足的技能,例如



那些团队最小值为 0，甚至更糟糕的，团队平均值为 0 的技能，或者那些团队平均值为 1 的技能，并在随后的几年内，提升其他技能的整体评分。

技能和资质	TA 最低要求	TTA 最低要求	TM Irit Kesef	TA Charlotte Wright	TTA James Malone	团队 最低要求	团队 平均要求
领域知识							
文本处理							
Windows 应用程序	1R	2R	2	2	1	1	1.7
Linux/UNIX 应用程序	D	D	1	1	1	1	1.0
Macintosh 应用程序	D	D	0	0	0	0	0.0
图像、图形和表格	1R	2R	2	2	0	0	1.3
文档管理							
Windows 应用程序	D	D	1	0	1	0	0.7
Linux/UNIX 应用程序	D	D	0	2	1	0	1.0
Macintosh 应用程序	D	D	0	0	0	0	0.0
文档交换							
Windows 应用程序	D	D	0	2	1	0	1.0
Linux/UNIX 应用程序	D	D	0	2	1	0	1.0
Macintosh 应用程序	D	D	0	1	0	0	0.3
打印							
颜色与带宽	D	D	1	2	1	1	1.3
激光、喷墨、其他	D	D	1	2	1	1	1.3
发布/装订	D	D	1	3	1	1	1.7
网站发布							
HTML	D	D	1	3	1	1	1.7
PDF	D	D	1	3	1	1	1.7
平均领域知识			0.7	1.7	0.7	0.5	1.0

图 10-7 使用清单进行缺口分析——领域技能

接下来让我们看看图 10-8 中的技术能力的缺口。这个结果与领域技能的评估结果相比看起来稍好一些。不过事实上，这个结果要比上一个好很多，因为对于技术的要求主要集中在 James，即技术测试分析师身上。另外，对于 Irit 这个测试经理只是基本了解（评分为 1）而不是熟练掌握（评分为 2）这些技能，我们也是可以接受的，因为项目团队的利益相关者对于测试人员掌握这一方面的技能的预期原本就不高。

因此，只要制定一个计划，向团队成员灌输一些 Mac 操作系统的基本知识，并让 Charlotte 熟悉一下各种编程概念即可。

技能和资质	TA 最低要求	TTA 最低要求	TM Irit Kesef	TA Charlotte Wright	TTA James Malone	团队 最低要求	团队 平均要求
能力与技术							
编程							
G/VB(3GL)	1R	D	1	0	3	0	1.3
Java/C++(OO)	1R	D	0	0	2	0	0.7
Shell脚本编程	2R	D	1	0	3	0	1.3
代码复杂度和度量	1R	D	1	0	2	0	1.0
操作系统							
Windows	1R	1R	1	1	2	1	1.3
Linux/UNIX	1R	1R	1	1	3	1	1.7
Max OS	D	D	0	0	0	0	0.0
网络/互连技术							
TCP/IP、FTP、RCP(网络架构)	1R	1R	1	1	2	1	1.3
浏览器(Firefox、IE等)	1R	1R	1	1	3	1	1.7
网络应用架构	1R	1R	1	1	3	1	1.7
网络硬件	1R	1R	1	1	1	1	1.0
系统和服务器							
网络/应用服务器	1R	1R	1	1	3	1	1.7
数据库服务器	1R	1R	1	1	2	1	1.3
平均技术能力			0.8	0.6	2.2	0.6	1.2

图 10-8 使用清单进行缺口分析——技术能力

### 10.3.1 动态测试团队练习

如果是在课堂上，那么 3~5 人组成一组讨论。如果只是您一个人，那么就只能自己做练习了。

使用上个练习中的技能列表对自己以及小组中的其他成员进行评估，根据评估的结果识别出团队中存在的缺口。概述解决这些缺口的方法。

一旦所有小组都完成了练习，那么小组之间可以开始讨论结果。

建议花 30 分钟的时间来做这个练习，包括讨论时间。

### 10.3.2 动态测试团队练习参考答案

这个练习的答案很大程度上取决于一个团队的特定情况。不过，一般一名测试经理可选的解决方法有如下几种：

- 招聘拥有所需技能的员工。然后将他作为这方面的专家，专门解决相关任务。



- 招聘拥有所需技能的员工。然后，在短期内将他作为这方面的专家，专门解决相关任务，从长期来看，可以安排他与其他团队成员进行交叉培训。
- 邀请拥有该技能的顾问。让顾问来解决那些相关任务（当需求是临时的、一次性的时候可以使用这种方法）。
- 邀请拥有该技能的顾问。让顾问来解决那些紧急并需要该技能的任务。在顾问离开前，安排顾问对其他团队成员进行交叉培训（顾问本身对这件事情可能不感兴趣，因此你应该预先提出，并在合同中明确标明培训是工作的一部分）。
- 将工作外包到具备该技能的公司。让外包公司解决这些相关任务（当外包公司具备规模经济或者该技能的学习曲线很长或很难掌握的时候，可以考虑外包）。
- 让某员工参加场外培训。然后，将他作为专家，专门解决需要该技能的任务。
- 让某员工参加场外培训。然后，在短期内将他作为专家，专门解决需要该技能的问题，从长期看，可以安排他与其他团队成员进行交叉培训。
- 对某些团队成员进行实地培训（通常有多名员工需要培训的时候才这么做）。然后将这些人作为专家，专门解决需要该技能的任务。
- 对所有的团队成员进行实地培训。在短期内，将那些培训成绩最好的员工作为专家，专门解决需要该技能的任务。从长期看，安排这些人对其他人员进行交叉培训。
- 与公司的其他部门进行交换，将拥有自己需要的技能的员工交换进来，将别人需要的员工交换出去。当然，在将你的员工交换出去的时候，你要确保这不会引起新的技能缺口。
- 在测试开始之前，要求开发团队向测试团队提供所需信息。

正如所看到的，上面的这些方法可能涉及测试团队的组织，甚至可能是公司层面的人力资源管理决策。因此并不是所有的测试经理都能用这些方法的。

## 10.4 组织的测试选择

### 学习目标:

(K2) 根据不同的组织类型进行外包/内部资源和外包/内部业务的选择。

在早期的计算机行业中，根本就没有什么特殊的、单独被称为“测试”的活动。开发人员对他们的代码进行调试，而这个过程中通常包含了一些单元测试的任务。但是这没什么作用。我的第一份工作是程序员，当时我们就是通过这种方法来确保软件质量的。事实证明，这将是一场持续的、千变万化的灾难。

尽管技术在更新换代并出现了敏捷生命周期模型，但这种方法依旧无法单独地被作为一种质量控制技术使用。那些古老的、尼安德特式的组织仅试图依靠这种方法来确保质量，但他们失败了。有些尖端的公司认为即使这种方法不奏效，他们最终依旧可以依靠一些华丽的编程语言或过程或工具来解决质量的问题，但他们还是失败了。软件质量问题是一个硬问题，Fred Brooks 认为它是“基本复杂度”而不是“偶发复杂度”的，而

对于基本复杂度的问题不存在什么奇妙的解决方法。<sup>2</sup>

在 20 世纪 80 年代末和 90 年代初出现了大批独立的测试团队，这是一种进步。在 20 世纪 80 年代末的时候我加入了一个独立的测试实验室，并在 90 年代初成为一名测试经理，我们向前跨了一大步。但是，我们又看到了一个新的问题，“将一切丢给测试的那伙人，让他们负责发布版本的质量”。现在，还是会发现很多组织有这种想法。

当质量是重要的时候，事实上质量总是很重要的，每个人都必须扮演一个角色。在先前的那些章节以及基础级大纲中，我们看到了很多能保证软件质量的过滤器。通常组织内的每个团队都需要采用其中几种过滤器。

有些过滤器，尤其是高等级测试，例如系统测试和验收测试，最好能有较高的独立性。而有些过滤器，尤其是低等级测试，例如单元测试，独立性低的时候效果更好。现在让我们来看看各种独立程度，然后才能做出我们的选择。

开发人员测试自己的代码称为自测试。当然，这里没有什么独立性。这种测试存在很严重的偏见，开发人员即使有足够的时间进行单元测试，他们也会遗漏掉一些很重要的缺陷，因为他们认为代码会像他们想象中的那样运行。但是，他们的观点可能与实际的需求不符。但是这么做也有益处，例如开发人员能很快地修复他们找到的缺陷，并能从技术的角度理解被测试系统。

开发人员互相测试代码而不是自己的代码称为同伴测试。结对编程是某些敏捷开发技术中所使用的方法，它是同伴测试的一种特殊形式，这种方法要求一个团队中的两个程序员一起开发代码、不断进行代码评审以及开发和执行单元测试，从而改进代码质量。虽然这里不存在明显的偏见问题，但是当两个人很紧密地一起工作时，很难说这其中的开发和测试有多大的独立性。另外，由于同伴互相测试代码，他们很可能选择不报告发现的缺陷，因此除非对这些人员进行特殊的培训，否则即使使用一些有用的缺陷度量元，也收集不到真实的数据。最后，由于一般的程序员很少有测试经验或经过培训，因此他们的想法主要是进行正面测试。当然，这么做还是有一些益处的，例如能很快地修复缺陷和能更好地从测试人员的角度去理解被测系统。

现在很常见的一个现象是开发团队内包含了一个或多个测试人员，因为很多敏捷开发技术的支持者建议这么做。作为质量保证过程中的一个部分，这并没有什么错，但是这种做法本身是危险的。可能会发生编辑和自编辑问题。

自编辑是指测试人员不向开发人员报告发现的问题，或只是进行非正式的报告，并且不使用缺陷追踪系统对缺陷进行正式的跟踪。自编辑就像是一个组织阻塞了自己的耳朵、眼睛之后瞎转悠，无法获得他们想要的质量。我们在第 3 章中曾讨论过，单独的缺陷度量元并不能说明什么，但是在任何恰当的、有意义的质量报告内都少不了这个度

---

2 Fred Brooks 一开始在他的“*No Silver Bullets*”一书中提到了基本复杂度和偶然复杂度的区别，不过这本书现在被作为一个章节收录在 *The Mythical Man-Month* 第二版中。简单地说，偶然复杂度是指由于食物或选择了错误的工具而引起的复杂度，例如在复杂的公路旅行的时候你忘了带一张地图或者你选用一辆雪地摩托进行沙漠旅行。基本复杂度是指问题内在的复杂度。试图检查并移除软件产品中潜在存在的无限的缺陷数目就是一个基本的硬问题。认为能够通过一些简单的变更就能解决这类问题，例如选一种新的软件开发生命周期或新的软件测试工具，这种想法本身就是一种问题，Brooks 在他的书中将这个问题称为过度乐观。



量元。

即使测试人员报告了缺陷，还是可能发生编辑问题。开发经理或项目经理不允许测试人员向其他利益相关者发布清晰、详尽的测试报告。此外，如果开发经理或项目经理关注短期目标，例如在预定的时间和预算内发布产品，测试人员的整体目标可能只是验证产品是否符合需求而已。最后，很多时候测试任务会被分配给像高级开发人员或者杂工等人员，而这些人通常还有其他任务。因此，测试总是在匆忙的情况下完成并且没有什么专业化可言。

虽然我们例举这些问题，但我还是认为有必要指定一些专业的测试人员为程序员提供咨询，这些测试人员本身可以是一个测试团队的成员。这些测试人员可以帮助程序员创建好的测试用例、自动化的测试套件、不断的集成—构建—冒烟测试过程中所需的设施等。我们曾为客户提供该项服务，并取得了很大的成功。但是，仅仅使用这种方法是不够的。

通常在验收测试或 Beta 测试中，业务人员、用户和技术支持人员会参与测试。这种方法的益处在于：它是真正独立的，并能诚实地向利益相关者报告测试结果。这些人关注的是系统是否能很好地协助他们完成工作，如果系统的质量很差，那么他们以后要受苦了。因此这种方法用于最后一个级别的测试是很有效的。

不幸的是，我们经常看到组织将这种方法用于系统测试，并且测试团队的技能是一维的。他们只关注领域知识，即使有技术能力，也非常有限。而且这些公司的高管轻视测试，说一些“哦，谁都可以测试。”之类的话。另外，测试人员经常以一种救火、不断打补丁直到起作用为止的态度在工作。

有些考虑周到的组织有独立的测试团队，测试团队负责执行系统测试、系统集成测试或者有时候是组件集成测试。用专业的测试人员对特定的测试对象进行测试有很多益处，与我们先前提到的方法相比，除了功能之外，它还能测试可用性、安全性和性能。虽然独立的测试团队有这么多益处，但是测试团队的形式化会降低测试的进度。糟糕的报告制度或管理也可能导致逆向的激励和不重视质量。

最后，在有些时候需要外部的测试组织进行测试。例如，在一些军事合同中经常会要求主承包商之外的独立团队对系统进行验证和确认。还有，可能会雇佣一个测试实验室对电子商务网站执行兼容性测试，以便节省购买设备的费用。这种方法的独立性是最高的。当然，测试和开发的责任完全明确之后可能导致信息的传递效率比较低，因此无法进行完全的测试。为了解决这些问题，组织必须明确定义需求以及团队之间通信的方法。

此外，还存在一个潜在的问题，即“谁来看守守卫”。我在本章开头的时候提到有一位防御设施供应商的员工希望获取一些无脑的、填空式的模板，用于创建测试计划、测试用例和测试规程等。我可以肯定地说，哪怕是程序员测试的时候存在偏见，也要比那些笨拙的测试人员做一些随意的、最简单的测试要好得多。任何使用外部测试组织的公司应该对组织的质量进行定期地审计，包括该团队的技能和专业性。

我列出了每种方法的益处和缺点。可以在多个质量过滤器中使用这些方法。这些益处和缺点能互补，因此使用具有不同独立性的方法来解决不同的任务是可行的。

### 10.4.1 混合使用不同独立性的方法

我在第8章讨论成熟度模型的时候说过，如果将事情简化为黑和白或二选一，你就会丢失很多重要的差别。在各种独立性方法的选择上人们也存在这个问题。人们会问，“我们应该使用程序员测试、用户测试、独立的测试团队、同伴测试、还是外部的测试实验室？”

这并不是一个单选题，而是一个多选题，应该根据特定的项目、产品和软件开发和测试过程组合使用这些方法。独立性是一种程度问题，而不是非此即彼的。

独立性是开发和测试两者之间的关系的一个属性。对于个人、团队或组织之中的任意两个实体，我们可以问“这两个实体之间的关系是什么，具有什么程度的独立性？”如果其中一个实体越是能自由地决定做什么，并且不用接受另外一个的命令，那么它的独立性就越高。

注意，我们这里并不是说可以无视他人的意见，而是它的活动无需得到他人的授权。这很重要，因为有些独立的测试团队会犯这样的错误，觉得他们应该做任何他们认为是正确的事情，而项目团队和组织只能被动地接受结果，他们认为这是独立性的一种表现。有着这种对抗式的、以为自己是质量警察的想法的独立测试团队通常都会以失败而告终。成功的独立测试团队会征求其他利益相关者的想法，在保持他们的独立性的同时，将为利益相关者服务和最大化项目团队和组织的利益为自己的目标。

我曾说过，增加测试的独立性并不是没有风险的。更高的独立性会导致更严重的孤立。它会降低测试人员对项目当前状态的了解程度，也会使开发人员丧失对质量的责任心。当然，这不是独立测试的必然结果，只是作为一名独立测试团队的经理他需要考虑缓解这种风险。

当然，降低测试的独立性也是有风险的。它会增加测试人员对项目的理解，但它也会引入一些有争议性的目标。降低测试的独立性会导致盲点的出现，例如什么才是真正的需求，它还会降低测试人员的专注和专业程度，并使得团队的技能组合变得不合理。

在某些时候，对软件开发模型的选择和其他项目决策会影响测试方法的选择。例如，如果使用的是敏捷开发模型，那么在开发团队内进行结对编程和测试应该是一种可选的方法。或者希望产品能兼容微软平台，那么需要选择一个完全独立于组织的微软兼容性测试实验室。

再次强调，可以混合使用上述不同程度独立性的方法。稍后会看到有一个组织成功地混合使用了这些方法。

如果将测试分成多份，分给了不同独立程度的测试组织，就应该遵守测试的基本规则，即你要明确定义每个组织的责任；每个测试级别、每个组织的测试目标。为此，可以和其他相关人员一起制定一个简洁、明确的测试方针文档，并获得管理层的许可。通过在软件生命周期的关键点设置一系列不同的过滤器，就可以在有限的时间和预算内获取最高的质量。



### ISTQB 术语

**测试独立性：**职责分离，有助于客观地进行测试。

## 10.4.2 外包独立测试

测试外包是一种外部的、独立的测试。它有各种不同的形式。一种是雇用外部的测试公司到本公司来提供测试服务（有时也称为内包）。另一种是选择一个离本公司比较近的外部测试公司。还有一种就是将测试外包到距离非常远的外部测试组织，可能跨越多个时区。

许多外包服务公司，包括 RBCS，能以上述任意一种方式为客户提供服务。我们有测试人员在客户的工作场所直接为他们工作的。我们也能在当地的某个场所，或国内任意的某个场所为客户提供测试服务。我们在新德里有一个分支机构，在那里我们也能提供测试服务。

在早些时候我们曾讨论过这个主题，现在让我们来回顾一下外包存在哪些问题：

- 外包测试团队与您的测试团队、开发团队可能存在文化上的差异。
- 项目团队和本地测试团队很难提供及时的、恰当的管理和指导，尤其是在那些混乱的、变更频繁的项目中。
- 缺少先见之明，本地项目团队和外包测试团队之间的通信存在很大的问题。糟糕的通信可能使上面提到的管理问题恶化。
- 没有仔细定义合同，导致知识产权的保护出现问题。即使合同定义得很好，在有些国家可以追索的补偿也很有限。
- 如果在外包的时候没有仔细选择外包测试供应商，那么这些外包测试人员的技能可能是有问题的。
- 严重的技能问题可能导致员工的高流动率。仔细地选择供应商和制定合同能有效降低这个风险。
- 由于发包的公司经常忘记对外包关系的管理也是有成本的，因此外包的成本预估通常都是不准确的。
- 最后，外包工作的质量可能更糟糕。

外包的距离越远发生这些问题的可能性越大。再次说明，我并不是说这些问题必然会发生，给项目带来困难，而是您需要使用我们第3章中提供的那些技术来管理和缓解这些风险。主要的解决方法还是在于仔细地挑选测试供应商并详细地定义合同。

## 10.4.3 混合的质量保证方法的案例分析

图10-9讲的是我们的一位客户是如何用不同独立程度的方法将各种质量保证任务，包括测试，集成到整个开发生命周期中去的。

首先，让我们看看生命周期。项目团队根据不同业务利益相关者的要求接手项目。

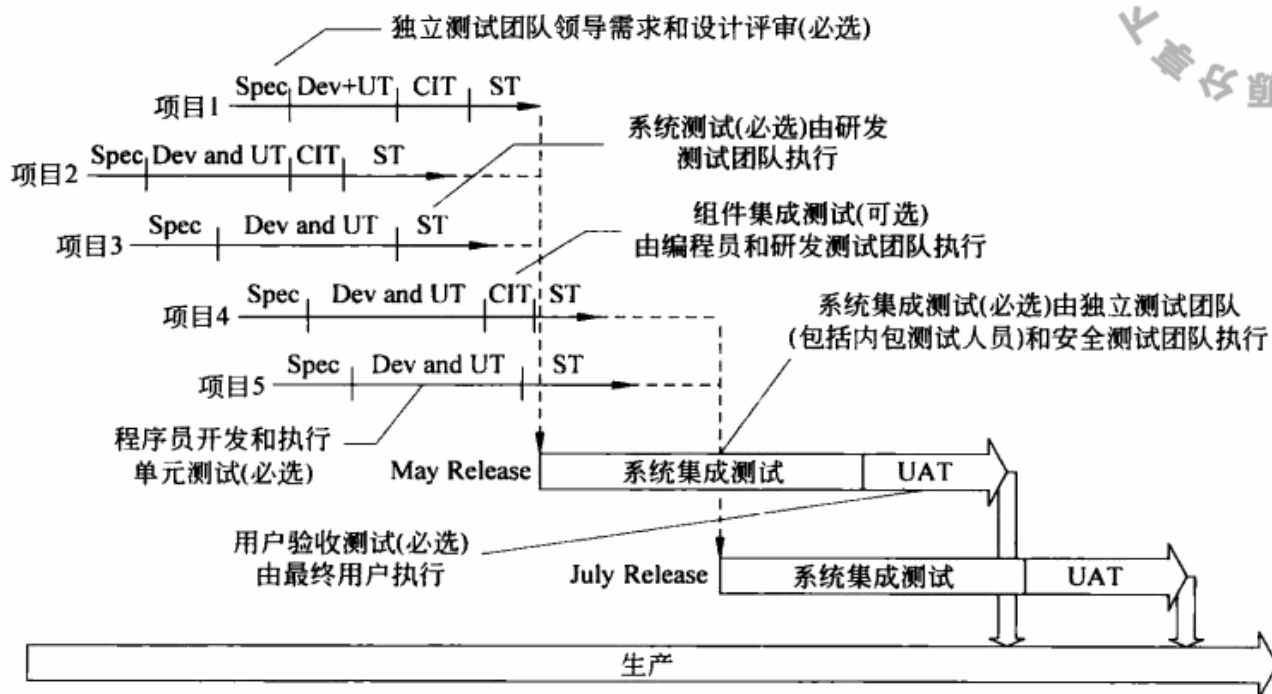


图 10-9 混合的质量保证方法

在部署前，这些项目必须集成起来接受测试，而后才能使用。约每两月就要进行一次集成、测试、运行。这种方法最大限度上降低了回归和交互风险。

对于每一个项目，随着项目的进行，需要举行一系列质量保证活动。每一个活动的独立程度不同、负责人不同，关注的重点也不同。

每个项目在一开始都要经历一个需求和设计规格说明阶段。独立的测试团队会对这些规格说明文档进行必需的评审。评审的参与者根据评审对象的不同有所变化，业务利益相关者和业务分析师参与需求评审，高级程序员、系统架构师、网络管理员和数据库管理员参与设计评审。

然后每个项目进入预集成测试阶段。换句话说，在与其他项目的代码集成之前，先要对该部分代码进行单独测试。在本阶段中，可以根据涉及的模块数目，选择是否要进行组件集成测试。开发团队可在测试团队的帮助下执行组件集成测试。内部的开发测试团队是由项目经理从开发团队中抽调人员组建的临时团队。因此，这种方法不是独立的，但是从技术上说开发人员对项目有很深入的了解。

预集成测试阶段必须包含系统测试。还是由开发测试团队执行这个测试。

计划将集成的项目必须执行系统测试，并测量一些项目度量，根据测量的结果决定该项目的质量是否已经达到了集成的要求。

独立的测试团队将对集成之后的软件进行系统集成测试，这是必须的。独立测试团队由内包测试人员组成，包括来自 RBCS 的测试人员。在系统集成测试阶段需要进行安全测试，这项测试由另一个独立的安全测试团队执行。

最后的质量过滤器是实际用户执行的用户验收测试，理想情况下是由那些专家级的用户执行的，他们对系统的业务相当地了解。

所以我们一共有 8 个质量过滤器：



- 需求评审。
- 设计评审。
- 代码评审。
- 单元测试。
- 组件集成测试。
- 系统测试。
- 系统集成测试。
- 用户验收测试。

这些过滤器的独立程度不同。

这种方法有效吗？几年之中这些项目的系统集成测试和用户验收测试的缺陷发现率一直保持在 99% 以上。这么高的缺陷发现的有效性部分是由于独立的测试团队和用户在最后的几次测试中只发现了很少的缺陷，大部分的缺陷在以前的测试中已经解决了。

因此，这种方法是有效的。事实上，如果你希望追求最高的质量，都应该采用这种方法。

## 10.5 激 励

### 学习目标:

(K2) 举例说明测试人员的正面和负面激励因素。

作为测试经理，需要激励我们的团队成员。应该如何激励测试人员呢？

赞誉。如果有人很好地完成了工作，那么就告诉他“你做得很好”。对于大多数人来说，公开地称赞他们的工作是一项主要的激励。不过，反过来说并不正确，即公开地指责别人的工作做得很糟糕并不能激励那个人。指责应该是私下的，并确保你的指责是建设性的，能指导测试人员进行改进。

管理授权。这里指的并不仅是您的授权。作为一名测试经理，应该努力提升团队并为你们的工作获取更高层次的授权。然后，与团队分享这些授权。

尊重。作为一名测试经理，应该在团队中培养一种互相尊重的氛围。还必须确保项目团队的其他成员对测试人员保持应有的尊重。当然，尊重是自己赢得的。

恰当地回报。包括金钱相关的，如薪水、绩效奖励和奖金。也包括一些非金钱的回报，如培训、职位提升。

当然，项目的某些因素可能限制对激励工具的选择。如果工作是不可能在规定时间内完成的，这就会让你的员工觉得绝望。另外，如果项目的进度很急，那么通常产品的质量会比较糟糕，因为每个人都在设法走捷径。

测试人员不应该独自决定缺陷是否延期，即使你已经经过深思熟虑并考虑了对客户的影响。不过，如果在不考虑影响的情况下就大规模推迟对缺陷的处理是一种负面的激励。

有时候项目处于关键时刻，测试人员就需要加班完成那些重要的任务。不过，如果

强制性加班，但又不做什么实际的事情，就是一种负面激励。

最后，让一名员工对某一部分的质量负责，但同时又有另一名员工在降低该部分的质量，成为替罪羊是一种负面的激励。

作为一名测试经理，无法阻止这些负面激励，但可以制定相应的解决方法。至少，你的团队会看到您正在努力地维护他们。没有什么事情比在一个不关心您的想法的经理手下工作更糟糕、更让人沮丧的了。

### 10.5.1 度量元和激励

不幸的是，度量元经常被误用，以致很多人认为度量元有负面激励的影响。不过，如果恰当使用产品度量元、过程度量元和项目度量元，这些度量元能证明测试的价值。

回想一下第3章中提到的那些度量元，可以测量团队的有效性，例如使用缺陷发现百分比测量缺陷的发现状况。可以测量团队的效率，例如，测量质量的成本。可以测量风险缓解的效果。还可以测量团队与其他相关团队或人员的沟通，例如，测量缺陷报告被拒绝的比率，以及询问利益相关者你的测试结果报告的质量。

为测试团队定义一个评估计划。可以使用我们第8章中讨论过的模型。在综合了测试团队、其他项目利益相关者，以及您自身的期望之后，为评估计划设定目标以及制定实现的过程。这些度量能帮你向管理层证明测试的价值和进展，同时也能向测试人员表明，尽管遇到很多令人沮丧的事情，但是作为一个团队整体，你们正在不断进步。

赞誉存在各种不同的形式。人们希望得到同伴和经理的尊重，希望同伴和经理认同他们的工作。他们想要得到晋升的机会、与同伴和行业平均水平相比而言比较公平的报酬，以及长远的发展。

一个不受尊重的测试团队是无法获得赞誉的。一个受尊重的人才能做出有意义的贡献并获得赞誉；这是一个良性循环，作为一名测试经理，您的工作就是要创建和维护这个循环。

为了创建和维护这个赞誉和尊重的循环，必须要能证明工作的价值。这需要用到度量元。那些坚持认为测试团队应该得到赞誉和尊重的测试经理，如果他们的依据仅仅是客观的工作报告，或更糟糕的，他们什么依据都没有，在这种情况下，组织的反应肯定会让他们失望的。

### 10.5.2 激励和负面激励评论的案例分析

一个对项目产生激励或负面激励的因素是从同伴那里得到的反馈。下面列出的是我遇到过的一些评论摘要，部分产生了激励作用，而部分则产生了负面激励作用。

让我们先来看看负面激励的评论：

- “现在我们要面对痛苦了。”这是一个程序员在电话会议上讲的话。他的意思是，既然开发人员因这些发现的缺陷而受苦，那么测试团队也应该承受同样的痛苦。不过，这种要求公平的想法非常诡异，因为测试团队正在承受并不是他们制造的缺陷所带来的痛苦。
- “即使我们已经在缺陷报告文档中标明这些缺陷是间歇性的或只有在测试环境中



才是可重复的，有些程序员还是以‘无法重复’为由将它们退回。”我向一位开发经理报告了开发人员在处理缺陷报告时存在的问题。这种现象对我们测试团队来说是非常让人沮丧的，因为很显然开发人员没有仔细阅读缺陷报告。

下面是一些激励的评论：

- “如果不曾让别人感到心烦过，那么您肯定不是在做这项工作”这是一个开发人员说的，他指的是另一位开发人员看到缺陷报告后的防御行为。能够得到开发团队内部的理解对于我们来说是一种激励。
- “从下周起，测试每周工作5天。”在经过一段每周工作6天的时期后，我告诉我的团队这种日子结束了。对此，他们都感到很高兴。
- “我会来的，我很喜欢龙舌兰！”在一个困难的项目结束后，一个经理同意参加测试团队的聚会。在社交聚会上，管理层的关注会让测试人员感到他们受到重视。重要的一点是，一些自然而然的评论对激励有很大的影响。

## 10.6 沟 通

### 学习目标:

(K2) 举例说明测试人员如何进行专业的、客观的、有效的沟通。在分析时应考虑相关风险和机遇。

对于大多数测试团队而言，有3种层次的沟通：

- 第一层，大多数时候我们会就测试产品文档在内部进行沟通，但有时候也会与别人进行沟通。包括对测试策略、测试计划、测试用例、测试总结报告和缺陷报告的讨论等。
- 第二层，我们会就评审文档的反馈信息进行沟通，无论是在测试小组内部还是外部，一般是同伴级的沟通。包括对需求、功能规格说明、用例和单元测试规格说明的讨论。
- 第三层，为了信息的收集和传播而进行沟通。不仅同级之间进行沟通，还有与经理、用户和其他项目利益相关者的沟通。如果测试结果显示项目的状态并不理想的话，在沟通的时候就需要小心处理。

无论是对测试人员，还是您这名测试经理，内部和外部的沟通都是一种很重要的专业技能。

作为一名测试经理，有效的沟通能帮助您达成目标，而无效的沟通只会妨碍您的工作。沟通有3要素：专业、客观、有效。每一次沟通，无论是内部的还是外部的，都应该要构建和维护他人对测试团队的尊重。在就测试结果、文档的反馈信息或其他不好的消息进行沟通的时候，需要一些交际手腕。

尤其是在测试执行的时候，测试人员压力很大，有时候就会显得情绪化。一旦发生这种状况，要告诉自己，应该专注于测试的目标，希望对产品 and 过程的质量进行改进，情绪化的沟通对达成这些目标没有任何好处。

和自己或者是同类人进行沟通是很容易的。换句话说，我们测试人员常常会使用一些缩略语来描述工作细节和发现，并用怀疑的眼光来看待事物。在与同伴讨论的时候，这是没问题的。不过，当目标是其他人的时候，就需要调整沟通方式。在与用户、项目团队成员、管理人员、外部的测试团队和客户等人进行沟通的时候，需要仔细考虑应该如何进行沟通、沟通什么、这种沟通是否能达成目标。

作为一名测试经理，我曾看到过一些无脑的电子邮件、缺陷报告或者走廊上的讨论对测试团队的声誉和信誉产生了恶劣的影响。因此，即使很忙，在沟通之前都应该想好怎么做。

图 10-10 描述的是一个测试沟通的例子。这是从一封发给供应商的电子邮件中摘录出来的一段文字，里面描述了新网站验收测试的结果。其中的第一段表明作者所要表达的是经过仔细地、深思熟虑地分析的，而不是那种经理每天可以收到十几封的考虑不周的邮件。这一段的意思是，“请认真对待这封邮件，因为我也是这么做的。”该段还引用了附件文档，以便读者能获取详细信息。

我花了几个小时来评审这个地方验收测试的当前状态。请查阅延迟处理的缺陷报告和第二轮测试状态报告附件。

下列问题必须解决以推动项目的进展：

- 消息和 UI 的合规性（例如缺陷 85、91、92、95、97）
- 没有简报的链接，违背了约定（见缺陷 98）
- 识别和解决内部的死链接（见缺陷 103）

上面的那些问题对不较真的读者来说也许是小事，但请理解我们的目标客户对任何错误都非常敏感……

为了让项目进展和进入这个关键的市场……我已经同意推迟处理部分第一轮验证测试发现的缺陷，以及供应商单方面撤掉的但是在宣传的产品功能中却有的功能。

但请注意，RBCS 同意推迟并不表明我们接受永久的推迟。

最后请注意，在第二轮测试中出现了 18 个新的缺陷。

图 10-10 描述验收测试状态的电子邮件

第二段总结在完成验收测试和开始部署前还需要做些什么，包括列表中的内容和结束句。

第三段解释了为什么要推迟部分缺陷。我不希望因为这封邮件而放弃 RBCS 应有的权利，这些问题稍后都是必须要得到解决的。

最后一段是很微妙的，我的意思是，对于发现一些新的问题我们感到很失望。

作为一名客户，我向供应商发送这样的一封邮件解释测试结果是恰当的。您会把这样的邮件发给开发人员吗？也许不应该。因为很重要的一点是，这封邮件的每一个词和每一句话都是为了达成预定的目标的。

## 10.7 认证考试模拟题

在每个章节结束之前，您可以尝试回答以下样题，加强对所学内容的理解，并准备 ISTQB 高级测试经理认证考试。



1. 您刚受聘担任测试经理，负责一个银行系统的集成测试、系统测试和验收测试。大部分银行系统是由以下几部分组成：大型机用于中央数据存储和提供服务、在支行或结算室部署 C/S 架构的基于 UNIX 的应用程序、而客户通过浏览器与银行产生交互。您从您的前任接手了一个测试团队。这个团队在发现缺陷和自动化回归测试上效率很高。您的前任由于对发布后的系统缺乏信心，以及他与组织内部其他经理的关系出现了问题，选择了离开公司。他原来选用的员工具有很丰富的测试同类或相似系统的经验，雇佣员工时他偏向于选择具有自学能力、有工作经验的人。  
基于以上描述，在对员工技能的 5 个主要方面（软件系统的技术知识、银行应用的业务知识、软件测试知识、人际交往能力、教育和培训）的调查中您最有可能发现些什么？
  - A 团队有广泛的银行领域的知识
  - B 团队专注于相关的技术知识
  - C 团队专注于测试相关的知识
  - D 团队有较强的人际交往能力
  - E 团队对测试基础相当了解
2. 继续上题，假如经理告诉您，可以再雇佣两个测试人员，还可以得到一笔培训预算为团队提供两个课程的培训，也可以利用新雇佣的测试人员为您的团队提供培训。请选择最佳的方案：
  - A 提供关于自动化测试的培训
  - B 提供针对测试理论的测试培训
  - C 提供银行应用的业务培训
  - D 雇佣一个有测试银行应用经验的测试人员
  - E 雇佣一个底层技术方面有经验的测试人员
  - F 雇佣一个表达能力比较强的测试人员
3. 假定您是一名可编程温控器项目的测试经理，该温控器用于控制中央加热、湿度以及空调系统（HVAC）。除了基本的 HVAC 控制功能外，这个温控器还有一个功能，通过浏览器能将数据下载到 PC 上，进行进一步分析。  
您决定将浏览器兼容性测试外包给一家大公司，这家公司能为测试、开发和技术支持提供离岸服务。通过对这家公司的调查，发现他们以前有为类似应用程序做过兼容性测试。对于外包组织，下面哪一项最有可能是正确的？
  - A 这个组织应该有所需的浏览器和 PC
  - B 这个组织应该有专业的测试人员
  - C 这个组织应该有一个稳定的测试团队
  - D 比起你自己的测试团队，这个组织的团队应该能做得更好
4. 有一个新的车型的发动机燃油控制模块需要升级。软件升级的目的是提高发动机燃油的效率。您是这个项目的测试经理。假如管理层对估算中提出的人员、时间和资源做出了保证，根据高级大纲中提到的概念，您将采用什么样的激励方案来鼓励团队成员？

- A 燃油效率提高 20%以上时给测试人员发奖金
  - B 在产品发布前发现了 90%的缺陷，就给测试人员发奖金
  - C 给发现缺陷最多的测试人员发奖金
  - D 在团队会议上批评做错事的个别测试人员
5. 假定您是一名银行项目的测试经理，有一个自动取款系统要升级到能够支持所有规定的信用卡，使持卡用户能在自动取款机上取钱。但不知道哪些是应该支持的信用卡，在需求规格说明中也没有说明。您如何用 E-mail 来沟通这个问题？请选择：
- A “在得到清单之前我们不会开始测试设计”
  - B “能告诉我什么时候能给我们要支持的信用卡类型清单吗？因为这个问题不澄清，测试设计有困难”
  - C “我再强调一遍，业务分析师的需求规格说明不完整。太典型了！”
  - D 什么都不说，只是描述一下这个问题造成的不良后果，稍后向项目经理解释为什么会出现延期



## 认证考试准备

“因为我很严厉，你们大概不会喜欢我。但是你们越是恨我，学到的就越多。”

——高级军事教练对美国海军新兵训练营的新兵们如是说，

出自 Stanley Kubrick 的越南电影《Full Metal Jacket》。

本书第 11 章介绍的话题与 ISTQB 高级测试分析师认证考试的准备相关。本章首先讨论 ISTQB 高级测试分析师的学习目标，这也是认证考试的基础。

本章有两节：

1. 学习目标。
2. ISTQB 高级认证考试。

如果您对参加 ISTQB 高级测试分析师认证考试不感兴趣，您可以跳过本章内容。

### 11.1 学习目标

所有高级大纲的认证考试都是基于学习目标的。学习目标说明了在参加认证考试之前您应该能够做什么。每个高级认证考试有自己的一套学习目标。在每章的每一节开始，我列出了高级测试分析师认证考试的学习目标。

学习目标有 4 个递进的难度。您必须达到这些学习目标才能正确回答后面给出的认证考试问题。认证考试会覆盖比较基础的记忆和理解，还有复杂的应用和分析。例如，要想答出如何创建测试计划这样的问题，您得记住并理解 IEEE 829 测试计划模板。不同于基础级认证考试，简单记忆和理解就足够解答问题了；在高级认证考试中，必须应用或者分析您所记忆理解的事实，这样才能给出问题的正确答案。

下面来看看高级认证考试 4 个级别的学习目标。我们分别用 K1、K2、K3 和 K4 表示 4 个级别，所以复习高级大纲时，要记住这些标记。

#### 11.1.1 级别 1：牢记（K1）

最低级别的目标，要求能够识别、牢记并且回忆某个术语或概念。K1 的关键字有：牢记、回忆、识别和认知。本级别学习目标的考查暗含在高级别目标的问题里。

例如，能够判断出如下语句是“失效”的定义：

- “没能为终端用户或者其他利益相关者交付服务”。
- “实际的组件或系统偏离期望的交付、服务或结果”。

这就意味着，您得记住 ISTQB 高级大纲使用的 ISTQB 术语，以及 ISTQB 高级大纲参考的标准，如 ISO 9126，IEEE 829。解答较高级别 K3 和 K4 涉及的问题，需要此级别的学习。

### 11.1.2 级别 2：理解（K2）

本级别学习目标要求您会推论或者解释相关主题的论点，并且可以总结、区别、分类和举例。K2 应用于实际情况，所以您要能够比较不同术语的含义。您也要理解测试概念。另外，理解测试步骤，比如解释任务的顺序也是要求之一。K2 的关键字如，总结、分类、比较、映射、对比、例证、解释、翻译、推断、定论和归类。

例如，您应该能够解释为什么测试应该尽早设计：

- 在缺陷造成的代价还比较小时，找出它们并移除。
- 先找出最重大的缺陷。

您也应该能够解释集成测试和系统测试的如下相同点和不同点。

- 相同点：测试多个组件，测试包括非功能性。
- 不同点：集成测试侧重接口和交互部分，系统测试侧重整体方面，比如，端对端处理。

以上意味着您必须理解 ISTQB 高级大纲使用到的 ISTQB 术语，以及 ISTQB 高级大纲参考的标准，如 ISO 9126，IEEE 829 的合理使用。解答较高级别 K3 和 K4 涉及的问题，需要此级别的学习。

### 11.1.3 级别 3：应用（K3）

本级别学习目标要求能够对概念或技术点在给定的条件下正确应用。这个级别适用于程序式知识。K3 不要求您评估软件应用或者为某个应用软件创建测试模型。提供一个模型，模型的覆盖要求，以及创建测试用例的步骤，这样的情况才属于 K3。K3 的关键字如，实施、执行、使用、按照流程和应用流程。

比如，您应能做到以下两点：

- 判断出有效或无效等价划分的边界值。
- 根据一般的测试用例创建流程，从给定的状态转换图（测试用例集合）选择测试用例，以覆盖所有转换。

以上表明，您应该能够应用 ISTQB 高级大纲中的技术回答特定的认证考试题目。此级别的学习包含 K1、K2 知识。

### 11.1.4 级别 4：分析（K4）

本级别学习目标要求能够把一个流程或者技术的相关信息分割成几个部分，以深入理解并区分事实与推论。K4 典型的问题是，通过分析一个文档、软件或者项目现状，提出合理的做法来解决问题或者完成任务。K4 关键字有：分析、区分、选择、结构、集中、



归因、解析、评估、判断、监控、协调、创建、合成、生成、假设、计划、设计、构造和生产。

例如，您应能够：

- 分析项目风险，提出预防和改正的行动以减轻风险。
- 判断事件报告中哪些部分是事实性的，哪些是由结果推导出的。

这意味着您要能够分析 ISTQB 高级大纲中的技术和概念，解答特定的认证考试问题。要求此级别的学习包含 K1、K2，甚至也包括 K3。

### 11.1.5 学习目标级别的由来

基础级和高级大纲的学习目标分类和级别划分，参考了 Bloom 于 20 世纪 50 年代定义的学习目标分类法。如果您参加过教育培训，您可能会发现这是非常标准的教育模式。

建议从如下实际角度来看这些级别：

- K1 要求记住基本事实、技术和标准，即使您可能还不理解。
- K2 要求理解事实、技术和标准，以及它们的关联，即使您还不能把这些应用到项目中。
- K3 要求能够应用事实、技术和标准于项目中，尽管您可能还不会使它们适合您的项目，或者选择当中最合适的。
- K4 要求能够分析项目中用到的事实、技术和标准，使它们适合项目或者选择最适合的。

从中可以看出，随着学习目标的升级，要求的能力也更高。高级别更注重应用和分析。

## 11.2 ISTQB 高级认证考试

和基础级认证考试一样，高级认证考试也是多项选择题。多项选择题包括三个部分。第一部分是主干，即题目的主体。主干可能包括文字、图或表格。第二部分是干扰项。如果您没有完全理解问题覆盖到的学习目标，您就可能把干扰项当成正确答案。第三部分是正确答案。

参加过基础级认证考试，您也许认为自己同样能通过高级认证考试。这几乎是不可能的。高级认证考试主要是 K3，K4 学习目标相关的问题。换句话说，您要能够应用和分析。K1 和 K2 学习目标，作为基础级认证考试的核心，将暗含于高级目标的问题里。

例如，基础级认证考试题目一般是这样的：

下面是测试计划的主要部分，请问下面哪项是符合 IEEE 829 的？

- |       |        |
|-------|--------|
| A 测试项 | B 探测影响 |
| C 目的  | D 预期结果 |

正确答案是 A，而 B、C 和 D 都是错误选项。这个题目要求回忆 IEEE 829 模板的主要章节。只有 A 在测试计划部分。C 和 D 分别在测试规程规格说明和测试用例说明中。B 是 ISTQB 术语表中的词汇。本题只是简单的回忆。

记忆很有用，尤其在学习开始时。但仅靠回忆事实并不能让您成为一名专家。我从1970年起是 AC/DC 乐队的主唱，深知除了记忆长长的歌词之外的能力的重要性。

高级认证考试的例题如下：

下面一段摘录于某个测试计划的测试项部分。

在系统测试期间，配置管理团队每周一早上九点前，把测试版本发给测试团队。每个测试版本要包括测试项移交报告。测试项移交报告描述该版本自动化编译的结果和冒烟测试的结果。一旦收到测试版本，如果冒烟测试是成功的，测试经理要把测试版本安装于测试实验室，然后周一早上就可以开始测试了。

如果测试团队没能收到测试版本，或者测试版本的冒烟测试不通过，或者版本没有安装，或者测试版本中没有移交报告，测试经理将立即联系配置管理团队的经理。如果问题不能在一个小时内解决，测试经理将汇报项目经理，然后有可能的话，继续上周版本的测试。如果测试版本安装失败，执行安装的测试分析师应写事件报告。

假设您是该项目的测试经理。您目前收到过两个可安装、可测试的版本。可是第三周的周一，您没有收到测试版本。下面哪个行动符合本测试计划？

- A 写事件报告，描述您发现版本未安装的时间和日期
- B 创建测试规程规范说明，描述如何安装测试版本
- C 给配置管理经理发 SMS 信息
- D 给项目经理和配置管理团队的经理发邮件

正确答案是 C。A、B 和 D 都是错误选项。A 错在不是版本没有安装，而是根本还没有到这一步。B 提到的测试规程也许对测试的安装有用，但是与本测试计划的推进没有关系。C 符合测试计划。D 不符合，原因是测试计划中提到了一小时缓冲。在这个一小时内，测试经理有可能把问题解决了，那么就不需要报告测试经理。另外，考虑到时间的重要性，电子邮件并不是好的方法。

可以看出，解答这类问题时需要分析，当然，了解 IEEE 829 的模板会有所帮助，如测试计划、事件报告、测试项转移报告和测试规程规格说明。事实上，如果您不了解标准，那么对术语可能会很迷惑。但是只是简单地了解 IEEE 829 标准并不能帮助您做出正确的解答，除非是偶然猜对的。

### 11.2.1 基于场景分析的考题

很多考题进一步加强难度时通常考虑场景。在场景考题里，会描述一系列的情况，然后给出 2 个、3 个甚至更多的考题。

例如，从测试计划和错过测试发布的场景，可能延伸出另外一组问题：

假设星期一的下午您终于收到一个测试发布版本了，当测试分析师尝试安装时，安装程序中的数据库配置脚本中途退出了。数据库服务器上的错误信息是用 Cyrillic 脚本显示的，尽管所选语言是美式英语。此时，数据库表格全部崩溃。使用测试中的应用程序，都会导致各种数据库连接错误信息（至少是用美式英语显示的）。

如下是一些可能的行动：

- I 通知配置管理团队的经理；



- II 通知项目经理;
- III 起草事件报告;
- IV 尝试重新安装;
- V 延迟测试;
- VI 继续尝试。

下面的哪一项描述了行动的正确顺序是合理的, 同时与测试计划的意图保持一致?

- A I, II, V
- B V, I, IV, III, I
- C VI, II, I, III, IV
- D II, I, V

答案是 B。A、C 和 D 是错误的。A 错误是因为没有起草事件报告, 而当安装失败时这是测试计划要求必须具有的。C 错误是因为一个有意义的测试不能在崩溃的数据库上进行, 因为在通知配置管理团队经理之前先通知了项目经理, 是因为在尝试重新复制失败之前起草了事件报告。D 错误是因为在通知配置管理团队经理之前先通知了项目经理, 没有起草事件报告。

可以看出, 对基于场景的考题, 在尝试答题之前认真分析场景非常重要。如果由于匆忙地阅读使您误解了场景, 您可以想象得出也许您错过了部分与之相关的问题。

让我重新回到学习目标这个问题。我说过考题包含 K1 和 K2 学习目标的——要求牢记和理解, 分别地包括在更高级别的 K3 或 K4 考题中。对 K1 学习目标还增加了额外的复杂程度: 它们没有被明确地定义。整个大纲, 包括使用的术语和提及到的标准, 都暗含覆盖了 K1 学习目标。

下面是高级大纲的一个摘录: 大纲的内容, 术语和所有标准的主要特征(目的)至少要记住(K1), 即便没有明确地在学习目标中说明。

因此如果您正在为高级认证考试做准备, 一定要多次仔细阅读高级大纲。

不仅要学习高级大纲, 您还要重新温习基础级大纲。下面是高级大纲的一部分摘录:

高级认证考试必须以高级大纲和基础级大纲为基础, 对于一些考题的答案, 可能要求运用高级大纲和基础级大纲的部分材料。高级大纲的所有部分和基础级大纲都是考查的对象。

注意上一段的第 2 句话意思是一个考题可能是高级大纲中两或三部分的交叉引用或者是高级某一部分与基础级大纲的交叉引用。因此, 去做一些基础级认证考试的模拟题并重新阅读基础级大纲是为高级认证考试准备的明智之举。

### 11.2.2 考题的演变

高级认证考试还在不断地演变中, 直到 2009 年底才完成并统一。即使哪一天高级认证考试指南完全制定, 还需要注意在高级大纲中包括一些隐晦的部分, 如下:

认证考试的形式是由 ISTQB 的高级认证考试指南制定。个别委员会可根据需要采用其他的认证考试机制。

我是根据 ISTQB 高级认证考试指南初稿来编写本章的。我假设大多数的 ISTQB 国

家委员会和认证考试委员会将遵循这些。不管怎样，根据高级大纲中的这一段，对于一些委员会创建的认证考试与高级 ISTQB 高级认证考试指南相偏离也是允许的，因此在这里需要说明。您需要向本国委员会或认证考试委员会咨询并确认。他们应该在网站上公布所创建的考题是根据 ISTQB 高级指南的哪一个版本。好了，通过阅读这些，您也许会紧张。不要这样！记住认证考试是测试您对高级大纲中学习目标的掌握程度。本书包含了这些重要的特征来帮助您为认证考试做准备，如：

- 您是否做了书中的练习？如果做了，那么您掌握了最难的学习目标级别，即 K3 和 K4 目标。如果没有，则需要回头去做了。
- 您是否做了本书中的模拟题？如果做了，那么已经熟悉大纲中多数学习目标的考题。如果没有，则需要回头去做了。
- 您是否去学习在本书中出现的 ISTQB 术语定义？如果是的，那么您已经熟悉了这些术语。如果没有，则需要重新学习 ISTQB 术语。
- 您是否学习了本书的每一章节和整个 ISTQB 高级大纲？如果是的，您了解了 ISTQB 高级大纲中的材料。如果没有，您需要复习 ISTQB 高级大纲，并再学习那些与大纲相应的最让您迷惑的部分。

我无法保证您一定通过认证考试，但如果您认真地学习了本书、ISTQB 术语和 ISTQB 高级大纲，您将有很大几率通过高级认证考试。

祝您考试顺利！也愿您能把高级大纲中的一些理念用于您的下一个测试项目中并获得成功。



## 参 考 书 目

### 高级教学大纲参考标准

British Computer Society. BS 7925-2 (1998), Software Component Testing.

Institute of Electrical and Electronics Engineers. IEEE Standard 829 (1998/2007), IEEE Standard for Software Test Documentation.

Institute of Electrical and Electronics Engineers. IEEE Standard 1028 (1997), IEEE Standard for Software Reviews.

Institute of Electrical and Electronics Engineers. IEEE Standard 1044 (1993), IEEE Standard Classification for Software Anomalies.

International Standards Organization. ISO/IEC 9126-1:2001, Software Engineering – Software Product Quality.

International Software Testing Qualifications Board. ISTQB Glossary (2007), ISTQB Glossary of terms used in Software Testing, Version 2.0.

US Federal Aviation Administration. DO-178B/ED-12B, Software Considerations in Airborne Systems and Equipment Certification.

### 高级教学大纲参考书籍

Beizer, Boris. *Black-Box Testing*. Wiley, 1995.

Black, Rex. *Managing the Testing Process (Second Edition)*. Wiley, 2002.

Black, Rex. *Critical Testing Processes*. Addison-Wesley, 2003.

Black, Rex. *Pragmatic Software Testing*. Wiley, 2007.

- Buwalda, Hans. *Integrated Test Design and Automation*. Addison-Wesley, 2001.
- Burnstein, Ilene. *Practical Software Testing*. Springer, 2003.
- Copeland, Lee. *A Practitioner's Guide to Software Test Design*. Artech House, 2003.
- Craig, Rick, and Stefan Jaskiel. *Systematic Software Testing*. Artech House, 2002.
- Gerrard, Paul, and Neil Thompson. *Risk-Based e-Business Testing*. Artech House, 2002.
- Gilb, Tom, and Dorothy Graham. *Software Inspection*. Addison-Wesley, 1993.
- Graham, Dorothy, Erik van Veenendaal, Isabel Evans, and Rex Black. *Foundations of Software Testing*. Thomson Learning, 2007.
- Grochmann, M. "Test Case Design Using Classification Trees." Conference Proceedings of STAR 1994.
- Jorgensen, Paul. *Software Testing: A Craftsman's Approach (Second Edition)*. CRC Press, 2002.
- Kaner, Cem, James Bach, and Bret Pettichord. *Lessons Learned in Software Testing*. Wiley, 2002.
- Koomen, Tim, and Martin Pol. *Test Process Improvement*. Addison-Wesley, 1999.
- Myers, Glenford. *The Art of Software Testing*. Wiley, 1979.
- Pol, Martin, Ruud Teunissen, and Erik van Veenendaal. *Software Testing: A Guide to the T Map Approach*. Addison-Wesley, 2002.
- Splaine, Steven, and Stefan Jaskiel. *The Web-Testing Handbook*. STQE Publishing, 2001.
- Stamatis, D. H. *Failure Mode and Effect Analysis*. ASQ Press, 1995.
- van Veenendaal, Erik, ed. *The Testing Practitioner*. UTN Publishing, 2002.
- Whittaker, James. *How to Break Software*. Addison-Wesley, 2003.
- Whittaker, James, and Herbert Thompson. *How to Break Software Security*. Addison-Wesley, 2004.
- Wiegers, Karl. *Software Requirements (Second Edition)*. Microsoft Press, 2003.



## 其他参考书籍

- Beizer, Boris. *Software Test Techniques*. 2e. Van Nostrand Reinhold, 1990.
- Bernstein, Peter. *Against the Gods*. Wiley, 1996.
- DeMarco, Tom, and Tim Lister. *Peopleware*, 2e. Dorset House, 1999.
- Dornich, Dieter. *The Logic of Failure*. Basic Books, 1997.
- Hetzel, William. *The Complete Guide to Software Testing*. Wiley, 1988.
- Jones, Capers. *Estimating Software Costs*. McGraw-Hill, 1995.
- Jones, Capers. *Software Assessments, Benchmarks, and Best Practices*. Addison-Wesley, 2000.
- Koomen, Tim. et al. *T-Map Next*. UTN Publishers, 2006.
- Levitt, Steven. *Freakonomics*. William Morrow, 2006.
- Tufte, Edward. *The Visual Display of Quantitative Information*, 2e. Graphics Press, 2001.
- Tufte, Edward. *Envisioning Information*. Graphics Press, 1990.
- Tufte, Edward. *Visual Explanations*. Graphics Press, 1997.
- Voas, Jeff, and Gary McGraw. *Software Fault Injection*. Wiley, 1998.
- Walsh, James. *True Odds*. Merritt Publishing, 1996.
- Walton, Mary. *The Deming Management Method*. Putnam Publishing Group, 1986.

## 其他参考资料

Holmes, Jeff. "Identifying Code-Inspection Improvements Using Statistical Black Belt Techniques." *Software Quality Professional*, December 2003, Volume 6, Number 1.

[www.risks.org](http://www.risks.org), the Risks Digest, for up-to-date information on how software defects affect real-world systems and their users.

[dictionary.com](http://dictionary.com), for standard English words.

[serc.carleton.edu/introgeo/socratic/](http://serc.carleton.edu/introgeo/socratic/), an explanation of the Socratic method of teaching (also applicable to writing test plans and test cases).

[www.rbcs-us.com](http://www.rbcs-us.com), to read various articles related to topics in this book.

## 附录 B

# HELLOCARMS 下一代房屋净值借贷系统

## 系统需求文档

本文档含有 RBCS 公司的专有和机密材料，严厉禁止任何未经授权的复制、使用或分发此全部或部分材料。本文档仅供 RBCS 员工、经授权的 RBCS 课程学员以及 Rex Black 书籍的购买者使用。

### I 目录

I	目录	354
II	版本	355
III	术语表	356
000	概述	357
001	非正式用例	359
003	范围	361
004	系统商业利益	362
010	系统功能需求	363
020	系统可靠性需求	368
030	系统易用性需求	369
040	系统效率需求	370
050	系统可维护性需求	371
060	系统可移植性需求	372
A	致谢	373

### II 版本

版 本	日 期	作 者	备 注	批准由/于
0.1	2007 年 11 月 1 日	Rex Black	初稿	
0.2	2007 年 12 月 15 日	Rex Black	第二稿	
0.5	2008 年 1 月 1 日	Rex Black	第三稿	



### III 术语表

术语 <sup>1</sup>	定义
房屋净值	房屋的直接市场价格与未付住房按揭贷款余额以及其他由房屋担保所产生的债务的差价。房主可通过减少未付住房按揭贷款余额和其他房屋担保债务来提高房屋净值。房屋净值也可随资产增值而提供。房主可根据房屋净值，通过房屋净值贷款、房屋净值信贷额度和反向住房按揭贷款（见下）的方式进行贷款
抵押贷款	借款人以资产为担保的贷款。如若借款人不能偿还贷款，出款方可合法要求获得抵押物，从而使该贷款得到保证
房屋净值贷款	在贷款之初，一次性发放给房主的有息贷款。房屋净值贷款为抵押贷款，以贷款人的房产净值作为担保
房屋净值信贷额度	预设最高额度，可供房主按需有息提取的可变贷款额度。房屋净值信贷额度使房主在需要时可提取由贷款人房产作为担保的抵押贷款
住房按揭贷款	用来购置房屋财产并以财产本身作为担保的法律贷款协议
反向住房按揭贷款	房主定期获得的住房按揭贷款，金额根据房屋净值计算。以使用房屋净值来获得养老金的方式最为典型。反向住房按揭贷款可使房主以房屋净值为担保，提取定期增长的贷款

#### 000 概述

第一版的房屋净值贷款、房屋净值信贷额度和反向住房按揭贷款服务系统（HELLOCARMS）已经实现了 Globobank Fairbanks 电话客服中心的客户经理（Globobank Telephone Bankers）接受客户对于房屋净值产品（贷款、信贷额度和反向住房按揭贷款）的申请。在第二版系统中，将会允许 Globobank 的业务合作伙伴以及用户通过互联网方式进行申请。

系统的概要配置如图 1 所示。

HELLOCARMS 系统本身是一组在 Web 服务器上运行的 Java 程序和各种接口。在受理申请过程中，数据库服务器提供存储空间，应用程序服务器则从 Web 服务器下载活动发送给客户。

#### 001 非正式用例

以下非正式用例应用于 HELLOCARMS 系统上的典型交易：

1. Globobank 电话银行客户经理在 Globobank 呼叫中心接听来自客户的电话。
2. 银行客户经理和用户进行电话访问，并在电脑上通过 Web 浏览器界面向 HELLOCARMS 系统录入信息。如果用户要求大额贷款或以担保高价值房产的方式借款，电话银行客户经理会将申请提交给高级电话银行客户经理，由他来决定是否继续处理该申请。

<sup>1</sup> 术语定义来源于 [www.dictionary.com](http://www.dictionary.com)。

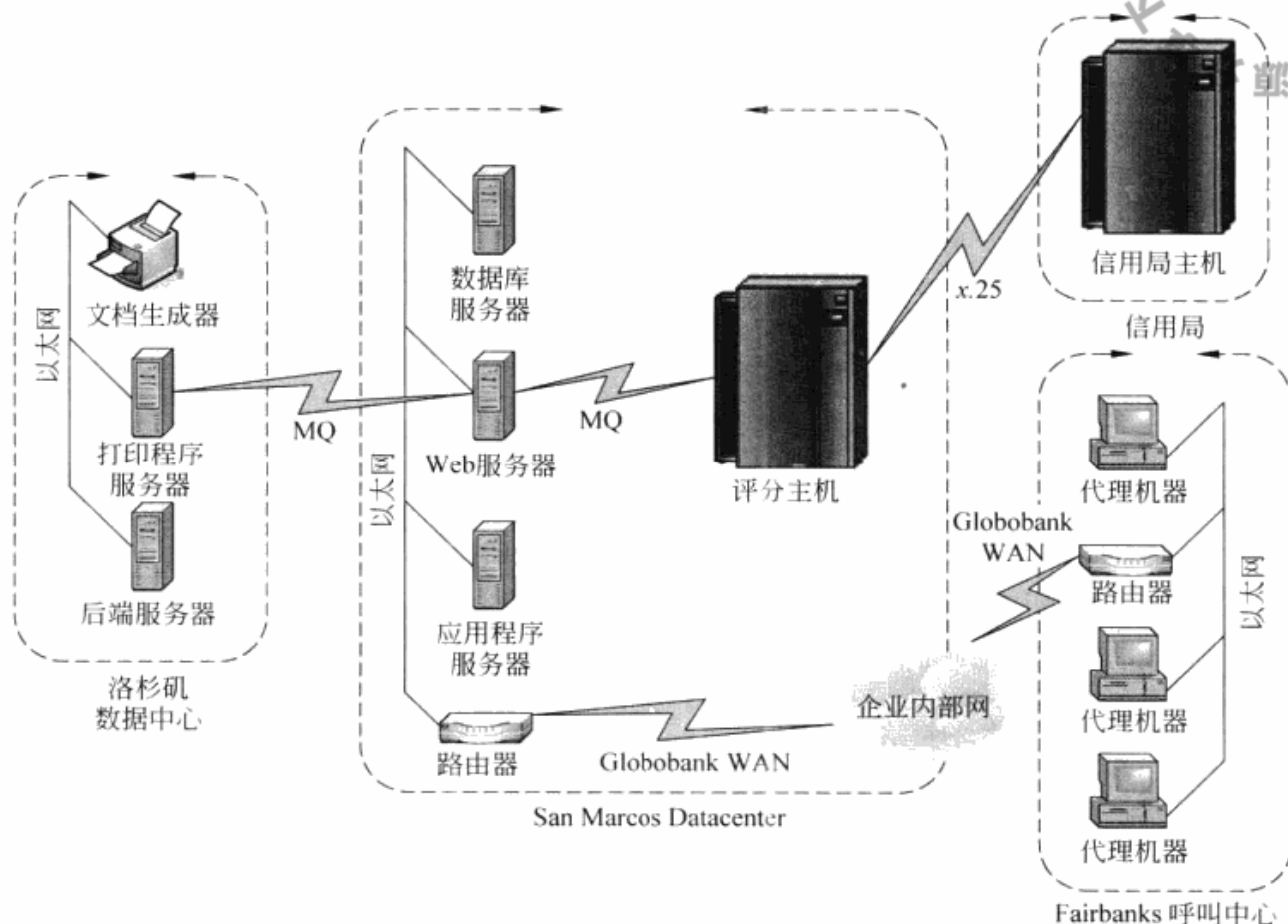


图1 HELLOCARMS 系统（第一版）

3. 一旦电话银行客户经理收集完用户的资料，HELLOCARMS 系统就会使用评分主机来确定客户的信用值。

4. 根据用户的所有信息，HELLOCARMS 系统显示电话银行客户经理可提供给客户的各种房屋净值产品（如果有的话）。

5. 如果客户选择其中一款产品，电话银行客户经理将有条件地批准产品。

6. 访问完毕。电话银行客户经理命令 HELLOCARMS 系统把贷款信息传输到洛杉矶的贷款文件打印系统（LoDoPS）。

7. 当发生下列事件时，HELLOCARMS 系统将收到来自 LoDoPS 系统的更新信息：

- a LoDoPS 系统发送文件给用户。
- b Globobank 贷款服务中心收到用户签字文件。
- c Globobank 贷款服务中心发出符合用户选择产品的支票或者其他材料。

一旦 Globobank 贷款服务中心给客户发出了所申请的资金或其他材料，HELLOCARMS 就完成了对申请的处理。系统将不再跟踪该客户后续的贷款相关活动。

HELLOCARMS 完成对申请的处理后，无论该申请是被银行拒绝，由用户取消，还是最后转变成有效的（active）贷款/信贷额度/反向住房按揭贷款，HELLOCARMS 应立即将该申请及相关的所有信息进行归档。



### 003 范围

HELLOCARMS 项目的范围包括:

- 从 5 个供应商中选择一个商业现货软件 (COTS) 解决方案。
- 与选定的应用程序供应商合作修改该方案, 以满足 Globobank 的需求。
- 提供浏览器前端给来自互联网、已有 Globobank 呼叫中心、外包 (非 Globobank 公司) 呼叫中心、零售银行业务中心或经纪人的贷款处理访问。但第一版的 HELLOCARMS 只提供来自 Globobank 呼叫中心 (确切的是 Fairbanks) 的访问。
- 开发连接到 Globobank 现有评分主机的接口, 用来根据贷款申请和 HELLOCARMS 特性给用户评分。
- 开发使用 Globobank 已有的承销和创始系统, 贷款文件打印系统 (LoDoPS) 以进行文档准备的接口。该接口允许 HELLOCARMS 系统在协助客户选择产品和提供初步批准之后, 转发预批准的申请 (贷款申请、信用额度申请, 或反向住房按揭贷款申请) 到 LoDoPS, 同时接口也允许随后至服务系统中跟踪申请的活动状态。
- 接收来自 GlobRainBQW 系统的客户相关数据用以生成对外邀请通知书, 通过电话、电子邮件及书面信件等方式发送给潜在的 (但非当前) Globobank 客户。

### 004 系统商业利益

与 HELLOCARMS 相关联的商业利益包括:

- 自动化当前的手动处理过程, 并允许通过互联网和电话中心的工作人员进行贷款咨询和申请 (包括从当前的呼叫中心和潜在的外包呼叫中心、零售银行业务中心和贷款经纪人)。
- 使贷款前端部分的处理时间从大约 30 分钟减少到 5 分钟。这将使 Globobank 的客户产品部门大幅增加贷款处理数量, 满足其商业计划。
- 由于 HELLOCARMS 将选择产品、决定申请人是否符合资格、推荐可替换的贷款产品, 并提供电话银行客户经理跟进的规则, 所以可降低对电话银行客户经理处理贷款申请所需技术门槛。
- 通过创始系统和文件预备过程, 提供在线申请状态和贷款的跟踪。这将使电话银行客户经理在处理用户申请过程中, 能迅速而准确地对用户咨询做出反应。
- 能够在同一环境中处理所有产品。
- 提供一致的方法, 决定是否提供贷款产品给客户, 若是, 提供什么贷款产品, 从而能够减少处理和销售错误。
- 允许基于互联网的客户 (在后续版本中) 访问 Globobank 产品, 选择喜爱的产品, 并在几秒钟内收到初步贷款审批意见。

HELLOCARMS 系统的业务发起人的目标是使约 85% 的客户咨询从系统中获利, 只有 15% 甚至更少的客户咨询需提交给更高级别的电话银行客户经理做特别处理。

## 010 系统功能需求

当软件在指定的条件下使用时，系统所提供的用以满足指定功能或默认需求功能的能力。

序 号	描 述	优先级*
010-010	适用性	
010-010-010	允许电话银行客户经理接收关于房屋净值贷款、信用额度和反向住房按揭贷款的申请	1
010-010-020	提供界面和脚本两种方式，以支持呼叫中心职员完成申请贷款处理	1
010-010-030	如果客户没有提供“你是怎么知道我们的”的识别代码，在申请处理过程中，系统将通过使用良好定义的来源分类下拉菜单搜集主要信息	2
010-010-040	提供数据验证，包括使用合理的用户接口（域）控制以及后端数据验证。输入域级验证的细节在单独的文档中进行描述	1
010-010-050	显示已有的债务，从而可以更好地考虑选定债务的偿还能力以进行债务合并。把需要偿还的已选债务按规定传递到 LoDoPS	1
010-010-060	允许电话银行客户经理、其他 Globobank 电话销售人员和合作伙伴访问未完成或被中断的申请	2
010-010-070	询问每个申请人当前是否与 Globobank 有业务往来，例如，拥有支票或储蓄账户。发送现有的 Globobank 客户关系资料到 Globobank 贷款申请数据存储中心（GLADS）	2
010-010-080	从开始到拒绝、婉拒，或接受（如果接受，还包括资金交付），可维护申请状态信息	2
010-010-090	允许用户终止申请。所有屏幕上提供终止功能	3
010-010-100	允许用户在一个单独的屏幕上说明，哪些（如果有）未清偿债务，将由客户使用正在申请的资金偿还。允许用户排除特定的债务和选择特定的债务。对于尚未清偿的债务，按规定发送相关信息到 LoDoPS，说明哪些是必须使用正在申请的贷款偿还的债务	3
010-010-110	如果用户要求偿还债务，则从该债务比例中去除债务的月度付款	3
010-010-120	如果客户没有贷款标识符，提供通过客户标识获得已有的申请请求	4
010-010-130	如果申请的贷款额超过\$500 000，指示电话银行客户经理将呼叫转移到高级电话银行客户经理；这类贷款需要有额外的管理核准	1
010-010-140	如果申请涉及的财产价值超过\$1 000 000，指示电话银行客户经理将呼叫转移到高级电话银行客户经理；这类申请需要有额外的管理核准	2
010-010-150	对于 Globobank 运营覆盖到的所有州、省和国家，提供本地和远程的电话销售支持	2
010-010-160	提供限定的合作伙伴专用的屏幕、标识、界面和品牌，以支持经纪人和其他业务伙伴	2
010-010-170	支持通过互联网方式递交申请，并保证未经训练的用户也能正确地进行申请操作	3
010-010-180	提供各种功能和屏幕，支持 Globobank 公司零售业务支行的运作	4
010-010-190	支持房屋净值申请的市场交易、销售和处理	1



续表

序 号	描 述	优先级
010-010-200	支持房屋净值信贷额度申请的市场交易、销售和处理	2
010-010-210	支持房屋净值反向住房按揭贷款申请的市场交易、销售和处理	3
010-010-220	支持组合金融产品（如房屋净值和信用卡）申请的市场交易、销售和处理	4
010-010-230	支持初次住房按揭贷款申请的市场交易、销售和处理	5
010-010-240	支持预审批申请的市场交易、销售和处理	4
010-010-250	支持灵活的定价策略，包括建议价格、短期价格等	5
010-020	<b>准确性</b>	
010-020-010	确定用户具备资格申请的各项贷款、信贷额度和/或反向住房按揭贷款，并把这些选项及其计算成本和条款提供给用户审阅。资格的确定必须与Globobank的信贷政策一致	1
010-020-020	根据房产的风险、信用评分、贷款与房产价值的比率，以及债务与收入比率，并基于评分主机的得分信息，确定客户的资格	1
010-020-030	在申请过程中，根据客户提供的申请资料评估月还款额度，将评估的月还款额度作为债务的一部分计入债务与收入比率计算，用于信用评分	2
010-020-040	对下列房产类型增加贷款收费： <ul style="list-style-type: none"> <li>● 1.5%，对出租房产（联式房屋、公寓和度假房）。</li> <li>● 2.5%，对商业房产。</li> <li>● 3.5%，对共管或合作房产。</li> <li>● 4.5%，对未开发房产。</li> </ul> 对其他支持的房产类型，单个家庭住宅，不增加贷款费用	3
010-020-050	以净值记录所有政府资产换置基金的收入（例如，美国的社会保障金），并在LoDoPS接口程序中转换为总值。[注：这是因为大部分政府资产换置收入是不收税的，但债务收入的计算中采用的是总值]	1
010-020-060	如果有的话，记录客户获得的额外收入（除工资、奖金、退休金之外）的周期（四舍五入到月）	3
010-030	<b>互操作性</b>	
010-030-010	如果客户在申请过程中提供“你是怎么知道我们的”的识别码，则从GloboRainBQW检索相关用户信息	2
010-030-020	接受共同申请（例如，合作伙伴、配偶、亲属等），评分主机将给所有申请人评分	1
010-030-030	指示评分主机删除共同申请者信用报告中重复的信用信息	2
010-030-040	允许用户在一个单独的显示屏幕上说明，哪些（如果有）是现有债务，应由用户使用正在申请的资金偿还。允许用户排除特定的债务，以及选择特定的债务。对于必须偿还的债务，发送一项规定条款到LoDoPS，说明哪些是使用借入资金偿还的债务	1
010-030-060	如果评分主机没有显示取消抵押品赎回权或破产的解除日期，但是客户表明抵押品赎回权已经恢复或破产已经解除，则需继续处理有关申请，并指示电话银行客户经理要求申请人提供解除的书面证明，发送到LoDoPS	3

序 号	描 述	优先级*
010-030-070	允许用户在一个单独的显示屏幕上说明, 哪些(如果有)是现有债务, 应由用户使用正在申请的资金偿还。允许用户选择特定的债务, 以及排除特定的债务。对于必须偿还的债务, 发送一项条款到 LoDoPS, 说明哪些是使用借入资金偿还的债务	3
010-030-080	以净值记录所有政府资产换置基金的收入(例如, 美国的社会保障金), 并在 LoDoPS 接口程序中转换为总值。[注: 这是因为大部分政府资产换置收入是不收税的, 但债务收入的计算中采用的是总值]	1
010-030-090	传递申请信息到评分主机	1
010-030-100	从评分主机接收评分结果和决策信息	1
010-030-110	如果评分主机出故障了, 对申请信息请求进行排队处理	2
010-030-120	一旦已验证的贷款发送到 LoDoPS, 则启动创始程序	2
010-030-130	传递所有被拒绝的申请到 LoDoPS	2
010-030-140	从 LoDoPS 接收申请状态的反馈信息	2
010-030-145	接收 LoDoPS 中贷款信息的变更(例如: 贷款数量、利率等)	2
010-030-150	支持计算机电话集成, 为本地电话促销活动和品牌业务伙伴提供个性化的市场和销售支持	4
010-040	安全性	
010-040-010	支持商定的安全要求(加密、防火墙等)	2
010-040-020	为每个申请跟踪“创建者”和“最后改动者”信息	1
010-040-030	允许外包的电话推销人员查看信贷级别, 但不能查看申请人的实际信用评分	2
010-040-040	支持通过互联网提交申请, 提供防止无意和有意安全攻击的安全措施	2
010-040-050	允许互联网用户浏览潜在贷款, 而无需透露他们的姓名、政府识别号码等个人信息, 直至在申请过程中必须需要个人信息才可执行时	4
010-040-060	对所有金融申请的处理, 支持欺诈侦测	1
010-050	与规范的合规性(功能标准/法律/规则)	
	[在随后版本中确定]	

\*优先级程度定义为

- 1 非常高
- 2 高
- 3 中等
- 4 低
- 5 很低

## 020 系统可靠性需求

当软件在指定的条件下使用时, 系统能够保持特定级别性能的能力。



序 号	描 述	优先级
020-010	成熟性	
	[在随后版本中确定]	
020-020	容错性	
	[在随后版本中确定]	
020-030	可恢复性	
	[在随后版本中确定]	
020-040	合规性（可靠性标准/法律/规定）	
	[在随后版本中确定]	

### 030 系统易用性需求

当软件在指定的条件下使用时，系统能够被用户和呼叫中心人员理解、学习、使用并认为有吸引力的能力。

序 号	描 述	优先级
030-010	易理解性	
030-010-010	支持通过互联网提交申请，并且能使未经训练的用户能进入申请程序	2
	[更多内容将在随后版本中确定]	
030-020	易学习性	
	[在随后版本中确定]	
030-030	易操作性	
030-030-010	提供完全用户定制化方式，适用于用户界面和用户为业务伙伴提供的文档，包括销售和市场信息的私人品牌，以及所有结账凭证	3
	[更多内容将在随后版本中确定]	
030-040	吸引力	
	[在随后版本中确定]	
030-050	与规范的合规性（可用性标准）	
030-050-010	遵从当地残疾人访问法规	5

### 040 系统效率需求

当软件在指定的条件下使用时，系统能够利用一定数量资源提供适当性能的能力。

序 号	描 述	优先级
040-010	时间行为	
040-010-010	用户可在 1 秒或更少的响应时间进行屏幕到屏幕切换。这一要求的测量应当从请求进入申请系统直到屏幕响应离开应用程序服务器，也就是说，不包括网络传输的延迟	2
040-010-020	5 分钟之内对递交的申请做出批准或拒绝	2



续表

序 号	描 述	优先级
040-010-030	在一个小时之内发起贷款，包括支付资金	3
	[更多内容将在随后版本中确定]	
040-020	资源利用	
040-020-010	每小时可处理最多 2 000 个申请	2
040-020-020	每小时可处理最多 4 000 个申请	3
040-020-030	可应付最多 4 000 个同时的（并发）申请提交	4
040-020-040	营运第一年支持总量达 120 万的经批准的申请	2
040-020-050	营运第一年支持总量达 720 万的申请	2
040-020-060	营运第一年支持总量达 240 万有条件地批准的申请	2
	[更多内容将在随后版本中确定]	
040-030	与规范的合规性（性能标准）	
	[在随后版本中确定]	

050 系统可维护性需求

系统能够被修正的能力。修正（Modification）包括改正（correction）、改进（improvement）或对于环境和需求以及功能性规约的变化所做出的软件适应性修改（adaptation）。

序 号	描 述	优 先 级
050-010	易分析性	
	[在随后版本中确定]	
050-020	易改变性	
	[在随后版本中确定]	
040-030	合规性（性能标准）	
	[在随后版本中确定]	

060 系统可移植性需求

系统能够从一个环境移植到另外一个环境的能力。

序 号	描 述	优先级
060-010	适应性	
	[在随后版本中确定]	
060-020	易安装性	
	[在随后版本中确定]	
060-030	可共存性	



续表

序 号	描 述	优先级
060-030-010	不应以任何非指定的方式与 Globobank 呼叫中心或数据中心的任何其他应用程序交互	1
	[在随后版本中确定]	
060-040	可替换性	
	无	
060-050	与规范的合规性	
	[在随后版本中确定]	

A 致谢

本文档来自于一个真实项目。RBCS 在此衷心感谢那些不希望留名的客户，感谢他们同意改编和发表该项目多个文档的匿名部分。

# 附录 C

## 模拟题答案

### 第 1 章

- |      |            |         |      |      |
|------|------------|---------|------|------|
| 1. A | 2. A, B, E | 3. D, E | 4. A | 5. C |
| 6. D |            |         |      |      |

### 第 2 章

- |            |      |      |      |
|------------|------|------|------|
| 1. A, B, D | 2. A | 3. A | 4. B |
|------------|------|------|------|

### 第 3 章

- |             |             |         |       |             |
|-------------|-------------|---------|-------|-------------|
| 1. C        | 2. B        | 3. A    | 4. D  | 5. A        |
| 6. C        | 7. B        | 8. A, C | 9. A  | 10. C       |
| 11. A, B, E | 12. B       | 13. D   | 14. C | 15. A, C, E |
| 16. D       | 17. A, C, D | 18. D   | 19. D | 20. A       |
| 21. B       | 22. A       | 23. C   | 24. D |             |

### 第 4 章

无

### 第 5 章

无

### 第 6 章

- |      |      |            |      |
|------|------|------------|------|
| 1. B | 2. D | 3. A, B, E | 4. B |
|------|------|------------|------|

### 第 7 章

- |      |      |
|------|------|
| 1. C | 2. A |
|------|------|

### 第 8 章

- |      |      |      |      |
|------|------|------|------|
| 1. C | 2. A | 3. D | 4. A |
|------|------|------|------|



## 第9章

1. A                      2. A, B, C                      3. D

## 第10章

1. B, C                      2. B, C, D, E                      3. A                      4. B                      5. B